



HOLLiAS-LEC G3系列小型一体化PLC

指令与功能块

北京和利时系统工程股份有限公司
杭州和利时自动化有限公司　PLC事业部











北京和利时系统工程股份有限公司
杭州和利时自动化有限公司　PLC事业部

地　址：北京市海淀区西三旗建材城中路10号
邮　编：100096
电　话：(010)82922200-2431/2430/2428/2425
传　真：(010)82928687/82923980
网　址：http://www.hollysys.com
技术支持邮箱：PLC@hollysys.com

版权申明

我们已核对本手册中的内容、图表与所述指令和功能块相符，但误差难以避免，并不能保证完全一致。同时，我们会定期对手册的内容、图表进行检查、修改和维护，恕不另行通知。

本手册的说明、图表、简单程序及应用实例完全出于举例说明的目的，我们对其都进行了测试，但因为软件版本的更新和各种应用有许多未知的变化和要求，我们不承担根据本手册或本手册中的实例而构成的实际应用产生的责任。

联系方式

北京和利时系统工程股份有限公司
杭州和利时自动化有限公司

北京市海淀区西三旗建材城中路 10 号
邮政编码：**100096**
电话：（**+86**）**010-82922200**
传真：（**+86**）**010-82928687**
网址：**http://www.hollysys.com**
技术服务邮箱：**PLC@hollysys.com**
版本：2005 年 12 月

前 言

HOLLiAS-LEC G3 系列 PLC 是和利时公司推出的新一代小型一体化 PLC，包括多种 CPU 模块和扩展模块。同时，和利时公司还推出了功能强大的 PowerPro 编程软件及丰富的指令与功能块。

HOLLiAS-LEC G3 系列 PLC 以其性能稳定、质量可靠、价格适中等优点，使之广泛应用于自动化领域的众多行业中，也赢得广大用户的好评。

包含内容

《HOLLiAS-LEC G3 系列小型一体化 PLC 指令与功能块手册》是和利时公司对其 G3 系列 PLC 包含的所有指令与功能块详细介绍的技术手册，主要包含以下信息。

- G3 系列 PLC 指令与功能块的概念
- G3 系列 PLC 的操作数与数据类型
- G3 系列 PLC 指令
- G3 系列 PLC 功能块
- G3 系列 PLC 部分实例

读者

本指令与功能块手册配合软件手册与硬件手册使用，适用于有一定 PLC 背景知识、掌握了 PowerPro 软件使用方法的工程师、编程人员。

使用本手册，需对 G3 系列 PLC 有一定的了解。

适用范围

本手册所讲述的所有指令与功能块适用于以下编程软件。

PowerPro1.0 中文版
PowerPro1.0 英文版
PowerPro2.0 中文版
PowerPro2.0 英文版

如何使用本手册

如果您已经熟练掌握 PowerPro 编程软件，直接通过目录在第三章至第七章查找需要的指令和功能块。

如果您刚刚掌握 PowerPro 编程软件，建议您首先阅读第一章 PowerPro 指令与功能块概述。

如果您对 PLC 所使用的操作数与数据类型不是很了解，建议您首先阅读第二章 PowerPro 操作数与数据类型。

附录 A 里包含指令与功能块速查表、功能块关联冲突速查表、硬件模块状态信息。

附录 B 里包含一些应用实例供使用者参考。

相关手册

《HOLLiAS-LEC G3 系列小型一体化 PLC 软件手册》
《HOLLiAS-LEC G3 系列小型一体化 PLC 硬件手册》

目 录

第1章 PowerPro 指令与功能块概述

和利时公司小型一体化 PLC HOLLiAS-LEC G3 的编程软件为用户提供了丰富的指令与功能块，绝大部分指令与功能块均可使用 LD、ST、FBD、IL 等方式进行调用，操作简单，使用方便。

本手册将详细讲述 PowerPro 所支持的指令与功能块。请注意，PowerPro1.0 版本包含本手册绝大部分指令与功能块，PowerPro2.0 包含本手册讲述的所有指令与功能块。

1.1 PowerPro 指令定义与分类

在可编程控制器中，使 CPU 完成某种操作的命令称为指令，指令的集合称为指令系统。指令系统是可编程控制器硬件和软件的桥梁，是可编程控制器程序设计的基础。

PowerPro 提供了丰富的指令，为了用户使用方便，按其代码存储位置分为两大类。

第一类：基本指令。此类指令代码存在于 PowerPro 软件内核之中，包括算术运算、赋值运算、布尔运算、移位运算、选择运算、比较运算、数据类型转换、初等数学运算、地址运算和调用运算，在使用时直接调用。

第二类：G3 指令。此类指令代码存在于 PowerPro 软件内核之外，主要存在于 Standard.lib 与 Util.lib 两个库文件之中，Standard.lib 包含字符串处理指令，Util.lib 中包含 BCD 码转换和位转换指令，在使用时需要添加相关的库文件。

注意

- 不论是基本指令还是 G3 指令，在使用的时候都无需声明。
- Standard.lib（标准库）在工程建立的时候自动载入到库管理器之中，无需用户载入；Util.lib（应用库）在使用时需要用户手动载入。
- Standard.lib（标准库）中除了 G3 指令还包含如下功能块：双稳态、触发器、计数器、定时器；Util.lib（应用库）中除了 G3 指令还包含如下功能块：高等数学运算、PID 控制、信号发生器、函数操纵器、模拟量处理。这些将在相关功能块示例中详细讲述。

1.2 PowerPro 功能块定义与分类

用于指挥CPU执行一定操作和完成一定功能的若干指令的组合称为功能块。对于用户来说，功能块的使用是透明的，只需载入包含该功能块输入输出接口的库文件（库名.lib）即可。其执行代码存储于不同的三个位置。为了用户使用方便，按执行代码的存储位置分为三类。

第一类：内部功能块。执行代码存在于“库名.lib”之中，比如高等数学运算、信号发生器等功能块的程序代码都存在于 Util.lib 之中，使用 PowerPro 软件即可打开 Util.lib 文件对该类功能块的执行程序进行修改。当程序下装到 PLC 之中，占用用户程序空间较大。

第二类：外部扩展功能块。执行代码存在于“库名.hex”文件之中，用户无法修改此类功能块的执行代码。使用时应保证 hex 文件的文件名与 lib 文件的文件名一致，且存在于同一目录下。当程序下装到 PLC 之中，占用用户程序空间较大。

第三类：外部 G3 功能块。执行代码已经存在于 G3 PLC 底层系统之中，用户无法修改此类功能块的执行代码。当程序下装到 PLC 之中，占用用户程序空间很少。

注意

- 所有的功能块在使用时都需要声明。
- 所有的功能块在使用时都只需载入“库名.lib”文件，系统会自动搜寻执行代码的位置。

1.3 PowerPro 库的定义与分类

编写 PLC 程序的过程中，经常会引用一些 G3 指令或者功能块，如字符串处理指令、触发器功能块、计数器功能块、PID 控制功能块等等。在 PowerPro 中我们把用来实现这些常用功能的 G3 指令和功能块集合起来进行分类，然后建立专门的库。

库是 G3 指令与功能块的集合，所有的库都包含“库名.lib”文件（包含 G3 指令和功能块的输入输出代码），调用 G3 指令和功能块只需载入相应的“库名.lib”文件。库中指令和功能块的执行代码存在于不同的位置，为了用户使用方便，我们按照与功能块类似的分类方法把库分为三类：

第一类：内部库。所包含 G3 指令与内部功能块的执行代码存在于“库名.lib”之中，可以使用 PowerPro 软件打开库文件对 G3 指令或者功能块的执行程序进行修改，用户也可以自己制作内部库。当程序下装到 PLC 之中，占用用户程序空间较大。

第二类:外部扩展库。所包含的 G3 指令与外部扩展功能块的执行代码存在于“库名.hex”文件之中，用户无法使用 PowerPro 软件打开此类库文件，修改此类功能块的执行代码。使用时应保证 hex 文件的文件名与 lib 文件的文件名一致，且存在于同一目录下。当程序下装到 PLC 之中，占用用户程序空间较大。

第三类：外部 G3 库。所包含的 G3 指令与外部 G3 功能块的执行代码已经存在于 PLC 底层系统之中，用户无法修改此类库所包含的执行代码。当程序下装到 PLC 之中，占用用户程序空间很少。

注意

- 使用 G3 指令或者功能块必须载入相关的库文件（库名.lib）。
- 标准库 Standard.lib 在工程建立时自动载入，无需用户再次载入。
- 内部库和外部扩展库只要载入，即使不调用其中的功能块，也会占用用户程序空间，建议只载入需要的库。

1.4 PowerPro 指令、功能块与库的关系

库是功能相近的指令与功能块的集合，其关系如表 1-4-1 所示。

表 1-4-1

		内部库	外部扩展库	外部 **G3** 库
代码位置	PowerPro	“库名.lib”文件	“库名.hex”文件	外部 G3 PLC
指令分类	基本指令	G3 指令		
功能块分类	无	内部功能块	外部扩展功能块	外部 G3 功能块

本手册 3－7 章将分别讲述基本指令、G3 指令、内部功能块、外部 G3 功能块和外部扩展功能块。

1.5 PowerPro 库安装方法

- 使用库时，需要保证相应的库文件存在于如下目录：PowerPro 安装目录\Library\
- 启动 PowerPro，选择“窗口/库管理器”，打开“库管理器”，在图 1-5-1 所示位置点击右键选择“添加库”。

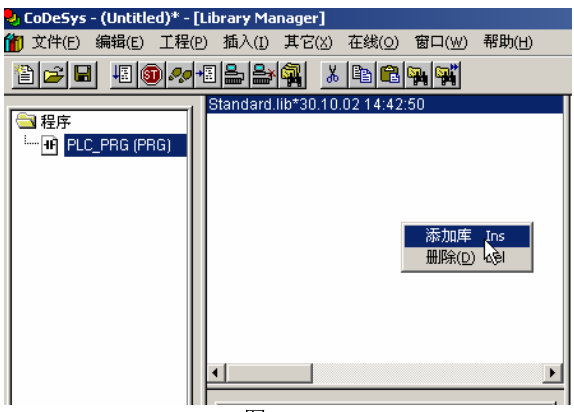


图 1-5-1

- 如图 1-5-2，选择需要的库文件，点击“打开”，不论哪种库只需要打开对应的*.lib 文件。

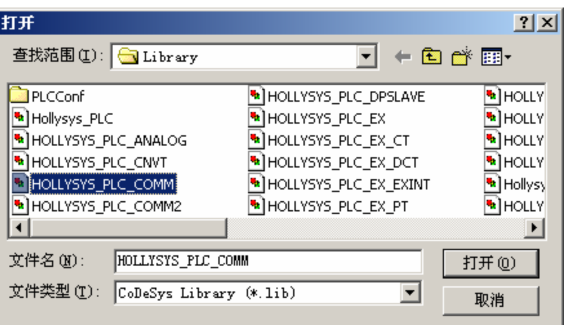


图 1-5-2

- 如图 1-5-3，上面选择的库被添加到列表中，该库所包含的功能块显示在图中鼠标位置。

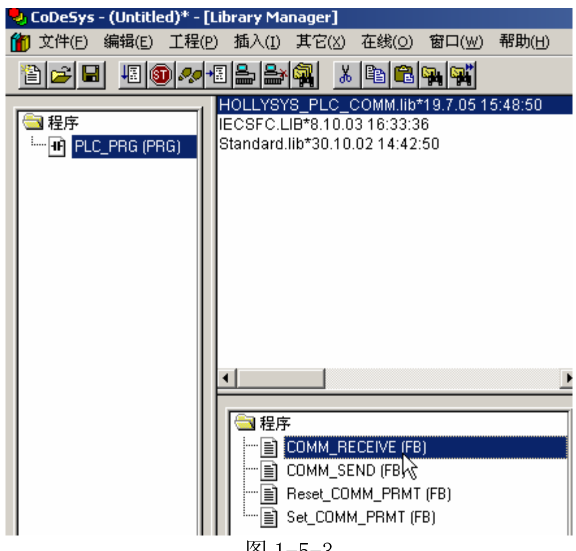


图 1-5-3

1.6 PowerPro 常用库

1.6.1 标准库 Standard.lib

Standard.lib 属于外部 G3 库，在工程建立时自动载入，无需用户再次载入，包含的 G3 指令与外部 G3 功能块如图 1-6-1 所示。

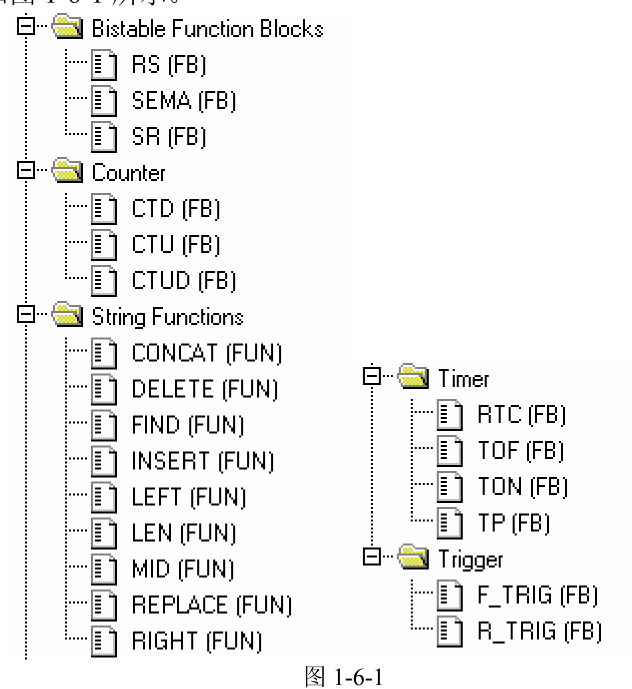


图 1-6-1

该库包含的G3指令与G3功能块的含义如表1-6-1所示。

表 1-6-1

双稳态功能块（Bistable Function Blocks）	计数器功能块（Counter）
RS（复位双稳态） SEMA（软件信号灯） SR（置位双稳态）	CTD（递减计数器） CTU（递增计数器） CTUD（递增递减计数器）
字符串指令（String Function）	**定时器功能块（Timer）**
CONCAT（合并字符串指令） DELETE（删除字符串指令） FIND（查找字符串指令） INSERT（插入字符串指令） LEFT（左边取字符串指令） LEN（取字符串长度指令） MID（中间取字符串指令） REPLACE（替换字符串指令） RIGHT（右边取字符串指令）	RTC（实时时钟） TOF（断电延时定时器） TON（通电延时定时器） TP（普通定时器） **触发器功能块（Trigger）** F_TRIG（下降沿检测触发器） R_TRIG（上升沿检测触发器）

1.6.2 应用库 Util.lib 和 Util_no_Real.lib

Util.lib 与 Util_no_Real.lib 属于内部库，使用时需用户载入。

◆ 应用库 **Util.lib**

Util.lib 包含的 G3 指令与内部功能块如图 1-6-2 所示。

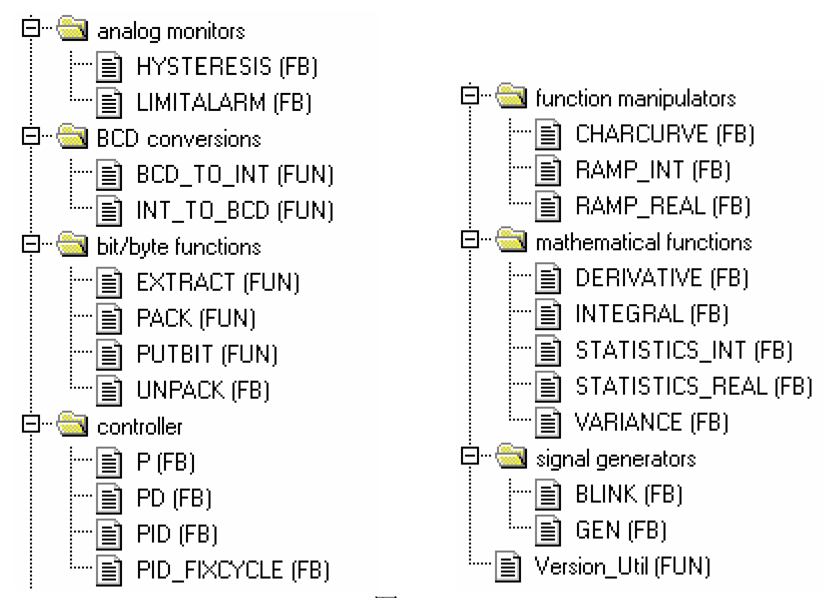


图 1-6-2

该库包含的G3指令与功能块的含义如表1-6-2所示。

表 1-6-2

模拟量监视功能块（analog monitors）	**BCD 码转换指令（BCD conversions）**
HYSTERESIS（滞后） LIMITALARM（上下限报警）	BCD_TO_INT（BCD 码转整型） INT_TO_BCD（整型转 BCD 码）
PID 控制器功能块（controller）	**高等数学运算功能块（mathematical functions）**
P（比例控制器） PD（比例微分控制器） PID（比例积分微分控制器） PID_FIXCYCLE （比例积分微分控制器，周期固定）	DERIVATIVE（微分） INTEGRAL（积分） STATISTICS_INT（整型统计） STATISTICS_REAL（实型统计） VARIANCE（平方偏差）
位转换指令（bit/byte functions）	**函数操纵器功能块（function manipulators）**
EXTRACT（位提取） PACK（位整合） PUTBIT（位赋值） UNPACK（位拆分）	CHARCURVE（特征曲线） RAMP_INT（整型限速） RAMP_REAL（实型限速）
信号发生器功能块（signal generators）	**库版本查看**
BLINK（脉冲信号发生器） GEN（典型周期信号发生器）	Version_Util（库版本查看）

◆ 应用库 **Util_no_Real.lib**

Util_no_Real.lib 包含的 G3 指令与内部功能块如图 1-6-3 所示。

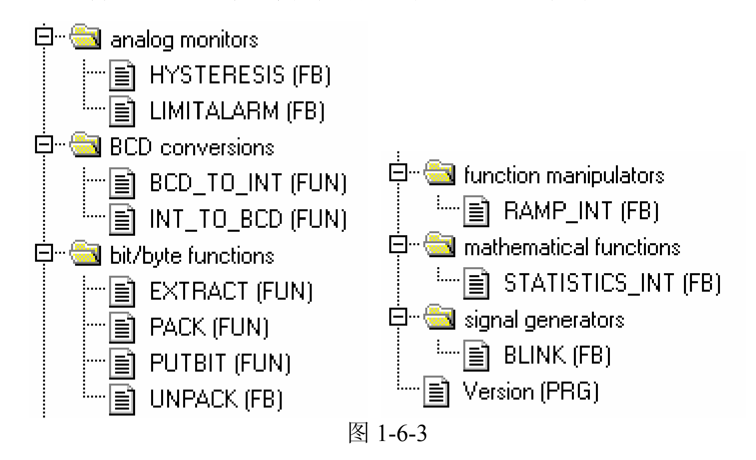


图 1-6-3

注意

➢ 库 Util_no_Real.lib 与库 Util.lib 的内容基本相同，只是没有与 Real 型变量有关的函数，编译下载后占用用户程序空间少。

1.6.3 系统库 SysLibCallBack.lib 和 SysLibC16x.lib

SysLibCallBack.lib 与 SysLibC16x.lib 属于外部 G3 库，其包含的 G3 指令分别如图 1-6-4 与图 1-6-5 所示。SysLibCallBack.lib 在工程建立时自动载入，无需用户再次载入，SysLibC16x.lib 使用时需用户载入，

◆ 系统库 **SysLibCallBack.lib**



图 1-6-4

该库可实现以下功能：

✓ 事件调用（SysCallbackRegister）

✓ 解除事件调用（SysCallbackUnregistrer）

◆ 系统库 **SysLibC16x.lib**

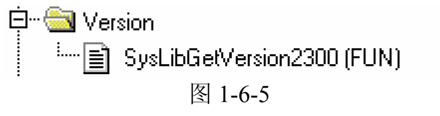


图 1-6-5

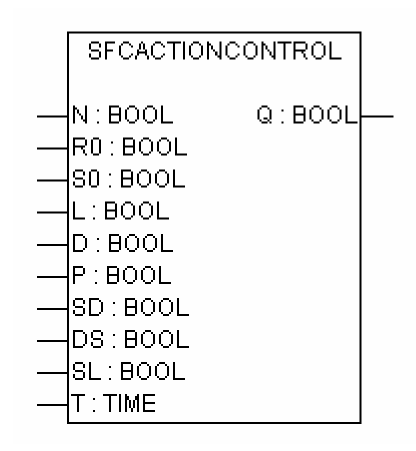
该库可实现以下功能：

✓ 读取版本信息（Version）：SyslibGetVersion2300可读取G3 PLC的版本信息。

注意

➢ 建立工程时 SysLibCallBack.lib 库自动载入，当使用系统事件时，会自动调用该库中的指令。

1.6.4 IEC 动作库 Iecsfc.lib



该库可实现以下功能

- ✓ SFCActionControl：IEC 61131-3 SFC动作控制

注意

- ➢ 该库的使用方法详见软件手册。

1.7 PowerPro 使用注意事项

- 上升沿使能，是指每当使能端由低电平变为高电平并一直保持高电平时，执行功能块的相关功能。
- 上电/下装后首次使能有效，是指当使能端第一次由低电平变为高电平并一直保持高电平，执行功能块的相关功能，以后再由低电平变为高电平将不会执行。如果要再次使用该功能块，需要重新下装或再次上电。
- 所有功能块只有当使能 EN 端、LOAD 端或 RSTLD 端在高电平时执行相应功能。
- 使用 Analog_IN、 Analog_OUT、DP_Slave 功能块时，Address 设置必须与 PLC 配置中相关模块的节点 id 一致，只有当 EN 输入使能后才启动功能块，进行相关的输入或输出。
- 使用 Set_COMM_PRMT 功能块设定自由口参数后，要想恢复原编程系统下装/调试功能，需将 RUN/STOP 开关拨到 STOP 位置，才可进行编程系统登录。
- 使用设置 Modbus 从站(SET_LOCAL_ADDRESS)功能块时,如果所需通讯参数不是 38400bps、8 位、无校验，则需要使用 Reset_COMM_PRMT 功能块设置相应的通讯参数，然后再使用 SET_LOCAL_ADDRESS 功能块设置站地址，程序下装前需要将 RUN/STOP 开关拨到 STOP 位置，程序下装后再将 RUN/STOP 开关拨到 RUN 位置运行程序。
- 只有使用 Set_COMM_PRMT 功能块设定 RS232 自由口参数后，才可使用 COMM_RECEIVE 功能块进行数据接收，使用 COMM_SEND 功能块进行数据发送。
- 只有使用 Set_COMM2_PRMT 功能块设定 RS485 自由口参数后，才可使用 COMM2_RECEIVE 功能块进行数据接收，使用 COMM2_SEND 功能块进行数据发送。
- %MB0～%MB99 为系统使用，用户不能再使用，可以读取，但不能写。
- 字型变量（%MW）地址必须定义在偶数地址，比如%MW200、%MW202、%MW204。
- 双字型变量（%MD）地址必须定义在偶数地址，比如%MD300、%MD304、%MD308。
- 在使用数学运算指令时，若输出数据定义的范围小于运算结果，则高位部分被删掉。
- 事件的调用需要使用 PowerPro2.0 以上版本，通过任务配置中的 System events 调用相应的事件，关于事件的使用请参考附录中的应用举例。
- 部分功能块因为使用了同一内部硬件，或者使用了相同的输入/输出端口，所以不可以同时使用，请参考附录中的 PowerPro 功能块关联冲突速查表。
- 若功能块声明在 RETAIN（掉电保持）区，该功能块的所有输入输出变量都要占用 RETAIN 内存区，所以建议不要在 RETAIN 内存区声明太多的功能块，以免超出 RETAIN 区大小限制。
- 在梯形图（LD）编程环境下，使能运算符调用与功能块调用的是两种不同的调用方式。其不同在于，如果采用使能运算符调用，当使能端低电平时,相应的指令或功能块不会执行，如果采用功能块调用，不论使能端低电平或高电平时，相应指令或功能块都会将使能端作为一个输入值来执行。

第2章　PowerPro 操作数与数据类型

2.1　操作数

在可编程控制器中，使 CPU 完成某种操作的命令称为指令，指令的集合称为指令系统。用于指挥 CPU 执行一定操作和完成一定功能的若干指令的组合称为功能块。指令系统是可编程控制器硬件和软件的桥梁，是可编程控制器程序设计的基础。

与计算机的操作指令类似，可编程控制器指令的基本形式也是由操作码和操作数组成。操作码表示 CPU 所要执行的操作类型和所要完成的操作功能。操作数表示 CPU 所要操作的对象和目的。常量、变量、地址和可能的函数调用都可以作为操作数。

2.1.1 常量

◆ 布尔常量

布尔常量只有两个：逻辑值 TRUE 和 FALSE（也可表示为 1 和 0），TRUE 等价于 1，FALSE 等价于 0。

◆ 时钟常量

时钟常量一般用来操作时钟，由“T#”（或“t#”）加上“时钟值”构成，时钟值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。

注意它们的正确顺序为 d、h、m、s、ms。

比如 T#12h38m16s 表示 12 小时 38 分 16 秒。

✧　举例

下面是正确的时钟常量：

T#18ms　　(*18 毫秒的一个时钟常量*)
T#100s12ms (*100 秒 12 毫秒的一个时钟常量，高单位允许超限*）
t#12h34m15s (*12 小时 34 分 15 毫秒的一个常量*)

下面是错误的时钟常量：

t#5m68s　　(*低单位不允许超限*)
15ms　　(*没有 T# *)
t#4ms13d　　(*顺序错误*)

◆ 日期常量

日期常量由“D#”(“d#”、“DATE#”或“date#”)加上“日期值”构成。

✧　举例

DATE#2005-05-06　　(*日期常量 2005 年 5 月 6 日*)
d#1980-09-22　　(*日期常量 1980 年 9 月 22 日*)

◆ 时间常量

时间常量用于存储时间，由“TOD#”（“tod#”、“TIME_OF_DAY#”或“time_of_day#”）加上“时间值”构成。

时间值的格式为：小时:分钟:秒（可以用实数形式输入秒）。

✧ 举例

TOD#00:00:00 (*时间常量为 0 点 0 时 0 分*)
TIME_OF_DAY#15:36:30.123 (*时间常量为 15 点 36 分 30.123 秒*)

◆ 日期时间常量

日期常量和时间常量合并起来称为日期时间常量，由“DT#”（“dt#”、“DATE_AND_TIME#”或“date_and_time#”）加上“日期时间值”构成。

✧ 举例

DT#1980-09-22-15:45:18 (*时间日期常量为 1980 年 9 月 22 日 15 点 45 分 18 秒*)
date_and_time#2001-03-09-00:00:00 (*时间日期常量为 2001 年 3 月 9 日 0 点 0 分 0 秒*)

◆ 数字常量

数字常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的 10 至 15 在十六进制中表示为 A 至 F。可以在数字中使用下划线连字符。

数字常量的数据类型可以是 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。

默认情况下，不允许“较大”的数据类型作为“较小”的数据类型使用。比如，DWORD 类型的常量不能简单地当作 INT 类型使用，必须使用数据类型转换指令进行转换之后才可以使用。

✧ 举例

14 (*十进制数 14*)
2#1001_0011 (*二进制数 1001_0011*)
8#67 (*八进制数 67*)
16#AE (*十六进制数 AE *)

◆ 实数常量

实数常量用十进制小数和指数来表示，遵循标准的科学计数法格式。

实数常量的数据类型是 REAL。

✧ 举例

7.4 (*实数 7.4*)
1.64e+009 (*实数 1.64e+009*)

◆ 字符串常量

字符串常量在两个单引号之间，可以包含空格和特殊字符。

✧ 举例

' Abby and Craig ' (*字符串 Abby and Craig *)
':-)' (*字符串:-)*)

在字符串中，当在字符中出现以$开始的复合字符时，解释为下面的形式：

字符串	代表含义
$$	$字符
$’	单引号
$L 或$l	输入行
$N 或$n	新行
$P 或$p	输入页
$R 或$r	换行
$T 或$t	Tab 键

2.1.2 变量

变量在 POU（Program Organization Unit）的变量表或者全局变量表中声明，在使用中应注意以下几点：

- 变量名不能包含空格和特殊字符，不能多次声明，不能和关键字使用相同的名字。
- 变量名不区分大小写。（如：VAR1、Var1 和 var1 表示相同的变量）
- 变量名识别下划线。（如：“A_BCD”和“AB_CD”被认为是两个不同的变量名）
- 变量名中不能有连续的 2 个下划线。（如：“A_ _B”是错误的变量名）

◆ 系统标志符

系统标志符是隐含声明的变量，它在每个特定系统中是不同的。使用命令“插入/声明关键字”打开输入辅助对话框，选择系统变量类别，这样可以找到系统中可用的系统标志符。

◆ 变量访问的语法

- ✓ 访问二维数组的元素：<字段名>[Index1, Index2]
- ✓ 访问结构变量：<结构名>.<变量名>
- ✓ 访问功能块和程序变量：<功能块名>.<变量名>

◆ 访问变量的地址位

- ✓ 在整型变量中，可以访问数据的每个数据位。
- ✓ 数据位附加在变量的后面，数据与数据位之间用“圆点”分隔，数据位从 0 开始编号。
- ✓ 下列数据类型都可以访问其数据位：SINT、INT、DINT、USINT、UINT、UDINT、BYTE、WORD 和 DWORD。
- ✓ 定义在 VAR_IN_OUT 数据区的变量，不能访问数据的地址位。

✧ 举例

```
a : INT;     (*定义整型变量 a*)
b : BOOL; (*定义布尔变量 b*)
...
a.2 := b;     (*将布尔变量 b 的值赋给整型变量 a 的第 2 位*)
```

◆ 出错处理

如果数据位大于变量数据位的宽度（比如上例中若 a.16 := b），则会给出下列出错信息：

Index '<n>' outside the valid range for variable '<var>'!

因为整型变量的数据位的范围是 0－15，a.16 超出了范围。

如果变量类型不允许（比如上例中若定义 a : REAL），则会给出下列出错信息：

Invalid data type '<type>' for direct indexing

因为实型变量不可以按位访问。

2.1.3 地址

◆ 地址格式

按照规定的地址格式显示内存中的地址。
格式如下：% 内存区范围 数据格式 地址

内存区范围	数据格式
I 输入区（Input） Q 输出区（Output） M 中间存储区（Memory location）	X 单个位 None 单个位 B 字节(8 位) W 字（16 位） D 双字（32 位）

✧ 举例

地址格式	对应地址
%QX7.5 或者%Q7.5	输出区的地址 7，第 5 位
%IW4	输入区的地址 4，1 个字
%QB7	输出区的地址 7，1 个字节
%MD48	中间存储区的地址 48，双字

◆ 内存位置

在 PowerPro 中，内存地址按照字节排列，从 0 开始，其大小与 PLC 型号有关。
比如 M 区（中间存储区）地址定义如表 2-1-1。

表 2-1-1

地址	字节定义	字定义	双字定义
0	%MB0	%MW0	%MD0
1	%MB1		
2	%MB2	%MW2	
3	%MB3		
4	%MB4	%MW4	%MD4
5	%MB5		
6	%MB6	%MW6	
7	%MB7		
……	……	……	……
4n	%MB4n	%MW4n	%MD4n
4n+1	%MB4n+1		
4n+2	%MB4n+2	%MW4n+2	
4n+3	%MB4n+3		

注意

➢ 字型变量（%MW）必须定义在偶数地址，比如%MW0、%MW2、%MW4、%MW6……%MW4n。每个字型变量占用 2 个字节型变量地址。
➢ 双字型变量（%MD）必须定义在偶数地址，比如%MD0、%MD4……%MD4n。每个双字型变量占用 4 个字节型变量地址或者 2 个字型变量地址。

◆ 数据存储格式

PowerPro 软件中数据存储格式如下面所示（以 M 区为例）。

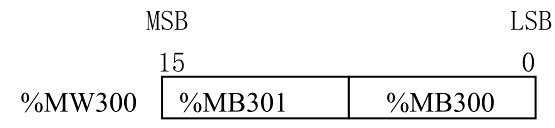
✓ 字节

MSB(最高有效位) LSB(最低有效位)

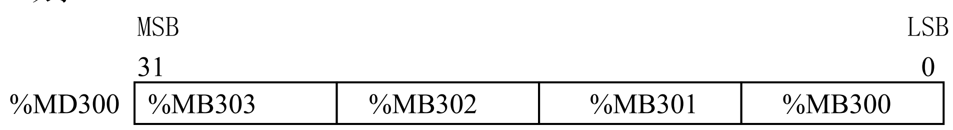
7 0

%MB300 | % MB300 |

✓ 字



✓ 双字



2.1.4 函数

在 ST 语言中，调用函数的返回值可以直接作为操作数使用。

✧ 举例

```
Result := Fct(7) + 3;   (*函数 Fct 的返回值加上 3，然后赋值给 Result *)
```

2.2 数据类型

编程时可以使用标准数据类型和用户自定义数据类型。数据类型用标识符来表示。数据类型规定数据占用内存空间的大小，并且规定存储于其中的数据种类。

标准数据类型包括布尔型数据、整型数据、实型数据、字符串型数据和时间型数据，可以使用 PowerPro 提供的数据类型转换指令使其相互转化。

用户自定义数据类型包括数组、指针、枚举和结构。

2.2.1 标准数据类型

◆ 布尔型数据类型

布尔型变量的标识符为 BOOL，其值为“TRUE”和“FALSE”（也可表示为 1 和 0）。TRUE 等价于 1，FALSE 等价于 0。

◆ 整型数据类型

整型数据类型的标识符包括 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 和 UDINT 等。每一种不同的数据类型的取值范围不同，整型数据类型的取值范围如表 2-2-1。

表 2-2-1

类型标识符	类型名称	数据下限	数据上限	存储空间
BYTE	字节型	0	255	8 Bit
WORD	字型	0	65535	16 Bit
DWORD	双字型	0	4294967295	32Bit
SINT	单整型	-128	127	8 Bit
USINT	无符号单整型	0	255	8 Bit
INT	整型	-32768	32767	16 Bit
UINT	无符号整型	0	65535	16 Bit
DINT	双整型	-2147483648	2147483647	32 Bit
UDINT	无符号双整型	0	4294967295	32 Bit

注意

➢ 当较长的数据类型转换为较短的数据类型时会丢失高位信息。

◆ 实型数据类型

实型数据类型也称为浮点型数据类型，用于表示有理数。实型数据类型的标识符为 REAL。REAL 型数据占用 32 位内存空间，即 4 个字节。

◆ 字符串型数据类型

字符串型数据可以包含任意多个字符，其标识符为 STRING。在声明时所输入的大小决定了存储变量所需要的内存空间，它是指字符串中字符的数量，可以用圆括号或方括号括起来。如果没有给出大小说明，则缺省大小为 80 个字符。

✧ 举例

str:STRING(35):='This is a String’;　(*声明一个包含 35 个字符的字符串*)

◆ 时间型数据类型

时间型数据类型用于处理时间，其标识符包括 TIME（缩写为 T）、TIME_OF_DAY（缩写为 TOD）、DATE（缩写为 D）和 DATE_AND_TIME（缩写为 DT）。

TIME 表示一个时间值，单位为毫秒，初始值为 0。

TOD 表示当天的时间，单位为毫秒，初始值为凌晨 0 点 0 分。

DATE 和 DT 表示当前日期和时间，单位为秒。初始值是 1970 年 1 月 1 日凌晨 0 点 0 分。

2.2.2 自定义数据类型

◆ 数组

一维、二维和三维数组属于基本的数据类型。在 POU 的变量表或者全局变量表中，都可以声明数组。数组的标识符为 ARRAY。

✓ 声明数组的语法

<数组名>:ARRAY [<L1>..<U1>,<L2>..<U2>,<L3>..<U3>] OF <基本数据类型>;

L1、L2 和 L3 表示字段范围的最小值，U1、U2 和 U3 表示字段范围的最大值。字段范围必须是整数。

✧ 举例

Card_game: ARRAY [1..13, 1..4] OF INT; (*定义一个整型的二维数组 Card_game*)

✓ 数组的初始化

在数组定义时，可以初始化数组中所有元素，也可以不进行初始化。

✧ 举例 1：数组的完全初始化

```
Arr1:ARRAY [1..5] OF BYTE:= 1,2,3,4,5;
Arr2:ARRAY [1..2,3..4] OF INT := 1,3(7) ;            （*即 1,7,7,7 的缩写形式*）
Arr3:ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3;  （*即 0,0,4,4,4,4,2,3 的缩写形式*）
```

✧ 举例 2：结构数组的初始化

```
TYPE STRUCT1:
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
END_ TYPE
ARRAY[1..3] OF STRUCT1:=(p1:=1;p2:=10;p3:= 3),(p1:=2;p2:=0;p3:=2),
                        (p1:=4;p2:=5;p3:=1);
```

✧ 举例 3：数组的部分初始化

Arr1:ARRAY [1..10] OF BYTE:= 1,2;

对于那些没有预先赋值的元素，按照基本数据类型的缺省初始值进行初始化。在此例中，元素[3]到[10]被初始化为 0。

✓ 数组的访问

在二维数组中访问数组元素，使用下列语法：

<Field_Name>[Index1,Index2]

✧ 举例

数组的访问：

Card_game[9,2]

注意

➢ 如果在工程中使用 CheckBounds 来定义函数，则可以自动进行数组越界错误检查。该函数名的关键字必须是 CheckBounds，CheckBounds 的程序如图 2-2-1 所示。

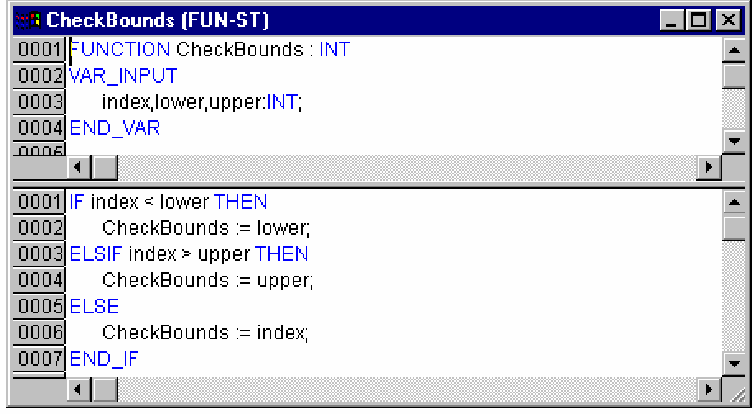


图 2-2-1

- ➢ 若工程中中没有上面所述的 CheckBounds 程序，在图 2-2-2 的例子中，A[B]应该为 A[10]，从而超出了数组 A 的最大上界值 7，程序编译时出错。
- ➢ 但因为工程中定义了上面的 CheckBounds 函数，所以执行图 2-2-2 的程序时，A[B]会被当作 A[7]来使用，将布尔量 TRUE 赋值给 A[7]，编译时不出错。

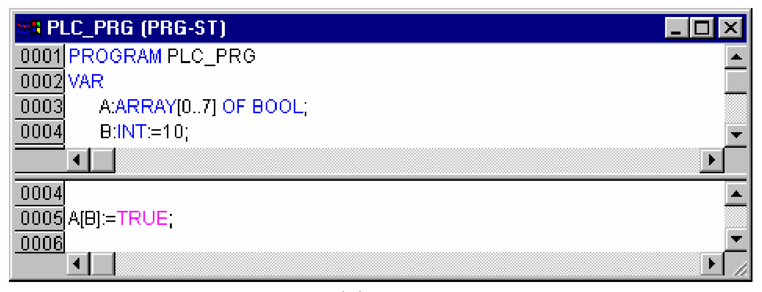


图 2-2-2

◆ 指针

当程序运行的时候，通过指针可以取得变量或功能块的地址。指针可以指向任何数据类型或功能块，包括用户自定义的数据类型。

✓ 声明指针的语法

<指针名>：POINTER TO <数据类型/功能块>；

通过取地址指令 ADR 可以将变量或功能块的地址取出。通过在指针标识符的后面增加取地址内容指令“^”可以取得指针所指地址的数据。

✧ 举例

程 序	含 义
pt:POINTER TO INT;	(*定义一个整型数据的指针 pt*)
Var_int1:INT := 5;	(*定义整型变量 Var_int1，使其等于 5*)
Var_int2:INT;	(*定义整型变量 Var_int2*)
pt := ADR(Var_int1);	(*取出 Var_int1 变量的地址，将地址值赋给 pt*)
Var_int2:= pt^;	(*将指针 pt 所指地址的值赋给 Var_int2，Var_int2=5*)

◆ 枚举

枚举是由一长串的数字常量组成的自定义数字类型。把这些常量称为枚举值。

只要枚举值声明在 POU 内，在整个程序范围内都可以被识别。建议将枚举创建在 PowerPro 程序左下角选项卡中的数据类型（Data types）里，通过在数据类型（Data types）上添加对象（Add Object）来创建一个新的枚举值。

枚举以关键字 TYPE 开始，以关键字 END_TYPE 结束。

✓ 声明枚举的语法

TYPE<标识符>:(<Enum_0>,<Enum_1>,…,<Enum_n>);
END_TYPE

注意

- ➢ 枚举值中可以包含标识符变量，初始化时，该标识符变量被初始化为该枚举中的第一个值。定义的枚举值与其他标准数据类型兼容，就像使用 INT 数据类型一样对其进行操作。可以把任何数字量分配给枚举值。如果枚举值没有被初始化，则从 0 开始递增进行初始化，当使用枚举值的时候，会重新调用有效的数据。

✧ 举例

程　序	含　义
TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10); END_TYPE	(*每个颜色的初始值是 Red=0，Yellow=1，Green=10*)
TRAFFIC_SIGNAL1: TRAFFIC_SIGNAL; TRAFFIC_SIGNAL1:=0; FOR i:= Red TO Green DO 　　i := i + 1; END_FOR	(*交通信号的值是 Red*)

➢ 相同的枚举值不能使用两次。

✧ 举例

```
TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
```

错误：red 不能同时被 TRAFFIC_SIGNAL 和 COLOR 使用。

◆ 结构

结构创建在 PowerPro 程序左下角选项卡中的数据类型（Data types）里，通过在数据类型（Data types）上添加对象（Add Object）来创建一个新的结构。

结构以关键字 TYPE 和 STRUCT 开始，以关键字 END_STRUCT 和 END_TYPE 结束。

✓ **声明结构的语法**

```
TYPE <结构名>:
STRUCT
   <变量声明 1>
   :
   <变量声明 n>
END_STRUCT
END_TYPE
```

<结构名>是一种可以在整个工程中被识别的数据类型，可以像标准数据类型一样引用，但不可以指定结构中变量的地址（即变量名之后不允许使用 AT 来指定该变量的地址）。

✧ 举例（定义名为 **Polygonline** 的结构）

```
TYPE Polygonline:
STRUCT
   Start:ARRAY [1..2] OF INT;
   Point1:ARRAY [1..2] OF INT;
   Point2:ARRAY [1..2] OF INT;
   Point3:ARRAY [1..2] OF INT;
   Point4:ARRAY [1..2] OF INT;
   End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

✧ 举例（初始化结构）

```
Poly_1:polygonline:= (Start:=3,3,Point1 =5,2,Point2:=7,3,Point3:=8,5,
                      Point4:=5,7,End :=3,5);
```

✓ **结构成员的访问：**

<结构名>.<结构成员名>

✧ 举例

如果结构名为“Week”，其中一个成员名为“Monday”，则可以用下面的形式访问：Week.Monday

第3章　PowerPro 基本指令

3.1　算术运算指令

3.1.1 ADD——加法指令

- 功能：两个（或者多个）变量或常量相加。
- 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程　序
LD	ADD (EN; 7, 2, 4, 7) → Var1=20
ST	Var1:=7+2+4+7;　（*结果 Var1 为 20*）
IL	LD　7 ADD　2,4,7 ST　Var1　（*结果 Var1 为 20*）
FBD	ADD (7, 2) → ADD (4) → ADD (7) → Var1=20

注意

➢ 时间变量也可以使用加法功能，两个时间变量相加得到一个新的时间。
例如：t#45s + t#50s = t#1m35s

3.1.2 MUL——乘法指令

- 功能：两个（或者多个）变量或常量相乘。
- 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程　序
LD	MUL (EN; 7, 2, 4, 7) → Var1=392

ST	Var1:=7*2*4*7; （*结果 Var1 为 392*）
IL	LD 7 MUL 2,4,7 ST Var1 （*结果 Var1 为 392*）
FBD	MUL: 7, 2 → MUL: 4 → MUL: 7 → Var1=392

3.1.3 SUB——减法指令

- 功能：两个变量或常量相减。
- 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程 序
LD	SUB: EN, 7, 2 → Var1=5
ST	Var1:=7-2; （*结果 Var1 为 5*）
IL	LD 7 SUB 2 ST Var1 （*结果 Var1 为 5*）
FBD	SUB: 7, 2 → Var1=5

注意

➢ 时间变量也可以使用减法功能，但时间不能有负值，请参照时间的加法。

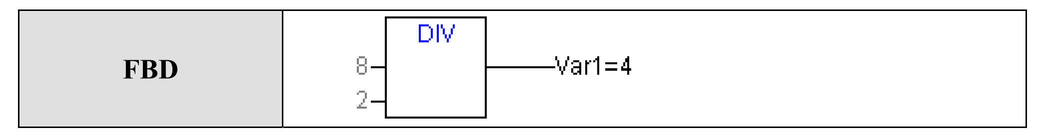
3.1.4 DIV——除法指令

- 功能：变量或常量相除。
- 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL。

✧ 指令使用举例

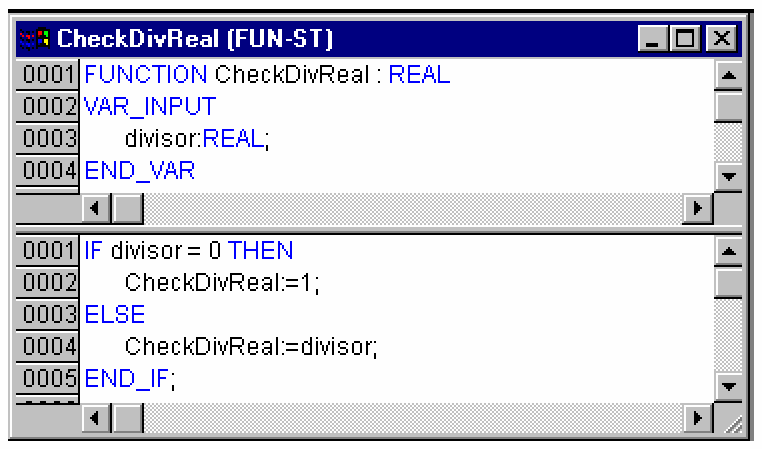
变量声明：Var1: INT;

POU 语言	程 序
LD	DIV: EN, 8, 2 → Var1=4
ST	Var1:=8/2; （*结果 Var1 为 4*）
IL	LD 8 DIV 2 ST Var1 （*结果 Var1 为 4*）

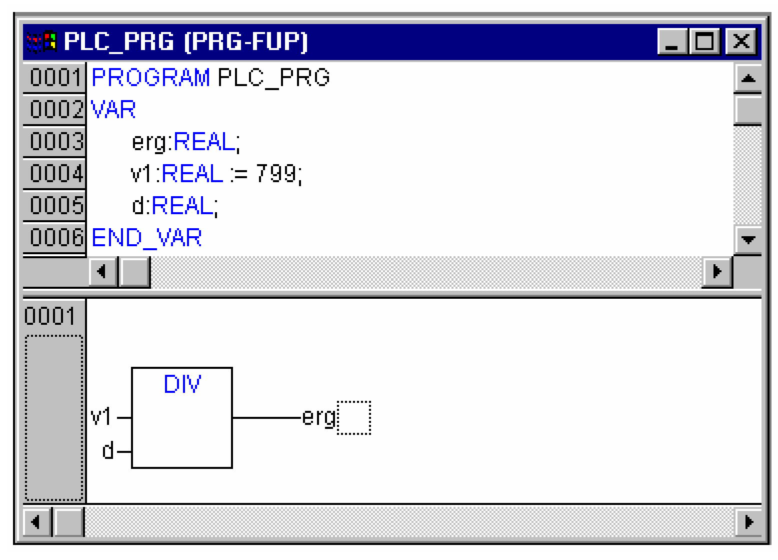
FBD	

注意

- 在工程中，当使用 DIV 指令时，如果使用 CheckDivByte、CheckDivWord、CheckDivDWord 和 CheckDivReal 等名称来定义函数，则可以用来检查除数的值，以避免被零除等事件的发生。函数必须采用上述所列出的名称。实现函数 CheckDivReal 的例子如下所示。



- 指令 DIV 使用函数 CheckDivReal 的输出值作为除数。下面的一段示例程序避免了被零除，进行除法时，若除数 d 为 0，则除数 d 自动从 0 变为 1，因此除法的结果为 799。



3.1.5 MOD——取余指令

- 功能：变量或常量相除取余，结果为两数相除后的余数，是一个整数。
- 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。
  - 指令使用举例

    变量声明：Var1: INT;

POU 语言	程　序
LD	MOD EN 9 2 Var1=1
ST	Var1:=9 MOD 2;　　(*结果 Var1 为 1*)
IL	LD　　9 MOD　　2 ST　　Var1　　(*结果 Var1 为 1*)
FBD	MOD 9 2 Var1=1

3.1.6 INDEXOF——索引指令

◆ 功能：在 POU 中执行索引指令，可以寻找内部索引，其输入为 POU 的名称，输出为 INT 的数据。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程　序
LD	INDEXOF EN POU2 Var1=38
ST	Var1:=INDEXOF(POU2);　(*结果 Var1 为 38*)
IL	LD　　POU2 INDEXOF ST　　Var1　　(*结果 Var1 为 38*)
FBD	INDEXOF POU2 Var1 (*结果 Var1 为 38*)

3.1.7 SIZEOF——数据类型大小指令

◆ 功能：取得数据类型所需字节数。

✧ 指令使用举例

变量声明：Arr1: ARRAY[0..4] OF INT;
Var1: INT;

POU 语言	程　序
LD	SIZEOF EN Arr1 Var1=10
ST	Var1:=SIZEOF(arr1)　(*结果 Var1 为 10*)
IL	LD　　arr1 SIZEOF ST　　Var1　　(*结果 Var1 为 10*)
FBD	SIZEOF Arr1 Var1=10

3.2 赋值运算指令

3.2.1 MOVE——赋值指令

◆ 功能：将一个常量或者变量的值赋给另外一个变量。
◆ 输入输出数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程 序
LD	MOVE EN 100— —Var1=100
ST	Var1:=100; (*结果 Var1 为 100*)
IL	LD 100 MOVE ST Var1 (*结果 Var1 为 100*)
FBD	MOVE 100— —Var1=100

3.3 布尔运算指令

3.3.1 AND——与指令

◆ 功能：变量或常量的相与运算。
◆ 输入输出数据类型：BOOL、BYTE、WORD 和 DWORD 等。

✧ 指令使用举例

变量声明：Var1: BYTE;

POU 语言	程 序
LD	AND EN 2#1001_0011— —Var1=2#10000010 2#1000_1010—
ST	Var1:=2#1001_0011AND2#1000_1010 (*结果 Var1 为 2#10000010*)
IL	LD 2#1001_0011 AND 2#1000_1010 ST Var1 (*结果 Var1 为 2#10000010*)
FBD	AND 2#1001_0011— —Var1=2#10000010 2#1000_1010—

3.3.2 OR——或指令

- 功能：变量或常量的相或运算。
- 输入输出数据类型：BOOL、BYTE、WORD 和 DWORD 等。
  - 指令使用举例

    变量声明：Var1: BYTE;

POU 语言	程 序
LD	OR EN 2#1001_0011 — Var1=2#10011011 2#1000_1010
ST	Var1:=2#1001_0011OR2#1000_1010 （*结果 Var1 为 2#10011011*）
IL	LD 2#1001_0011 OR 2#1000_1010 ST Var1 （*结果 Var1 为 2#10011011*）
FBD	OR 2#1001_0011 — Var1=2#10011011 2#1000_1010

3.3.3 XOR——异或指令

- 功能：变量或常量的异或运算。
- 输入输出数据类型：BOOL、BYTE、WORD 和 DWORD 等。
  - 指令使用举例

    变量声明：Var1: BYTE;

POU 语言	程 序
LD	XOR EN 2#1001_0011 — Var1=2#00011001 2#1000_1010
ST	Var1:=2#1001_0011XOR2#1000_1010 （*结果 Var1 为 2#00011001*）
IL	LD 2#1001_0011 XOR 2#1000_1010 ST Var1 （*结果 Var1 为 2#00011001*）
FBD	XOR 2#1001_0011 — Var1=2#00011001 2#1000_1010

3.3.4 NOT——取非指令

- 功能：变量或常量的取非运算，逐位取非。
- 输入输出数据类型：BOOL、BYTE、WORD 和 DWORD 等。
  - ✧ 指令使用举例
    变量声明：Var1: BYTE;

POU 语言	程　序
LD	NOT EN 2#1001_0011 — Var1=2#01101100
ST	Var1:= NOT2#1001_0011 （*结果 Var1 为 2#01101100*）
IL	LD　　2#1001_0011 NOT ST　　Var1　　（*结果 Var1 为 2#01101100*）
FBD	NOT 2#1001_0011 — Var1=2#01101100

3.4 移位运算指令

3.4.1 SHL——左移指令

- 功能：对操作数进行按位左移，左边移出的位不作处理，右边自动补 0。
- 输入输出数据类型：BYTE、INT、WORD、DWORD 等。
  - ✧ 指令使用举例
    变量声明：Var1: BYTE;
    Var2: INT;

POU 语言	程　序
LD	SHL EN 2#1001_0011 — Var1=2#01001100 2
ST	Var1:=SHL(16#45,2)　（*结果 Var1 为 16#14*） Var2:=SHL(16#45,2)　（*结果 Var2 为 16#0114*） 注意：上面例子中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
IL	LD　　16#45 SHL　　2 ST　　Var1　　（*结果 Var1 为 16#14*）
FBD	SHL 2#1001_0011 — Var1=2#01001100 2

3.4.2 SHR——右移指令

- 功能：对操作数进行按位右移，右边移出的位不作处理，左边自动补 0。
- 输入输出数据类型：BYTE、INT、WORD、DWORD 等。
  - 指令使用举例

    变量声明：Var1: BYTE;

    Var2: INT;

POU 语言	程　序
LD	SHR EN 2#1001_0011 — 2 — Var1=2#00100100
ST	Var1:=SHR(16#45,2)　（*结果 Var1 为 16#11*） Var2:=SHR(16#45,2)　（*结果 Var2 为 16#0011*）
IL	LD　16#45 SHR　2 ST　Var1　（*结果 Var1 为 16#11*）
FBD	SHR 2#1001_0011 — 2 — Var1=2#00100100

3.4.3 ROL——循环左移指令

- 功能：对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。
- 输入输出数据类型：BYTE、INT、WORD、DWORD 等。
  - 指令使用举例

    变量声明：Var1: BYTE;

    Var2: INT;

POU 语言	程　序
LD	ROL EN 2#1001_0011 — 2 — Var1=2#01001110
ST	Var1:=ROL(16#45,2)　（*结果 Var1 为 16#15*） Var2:= ROL (16#45,2)　（*结果 Var2 为 16#0114*） 注意：在循环左移过程中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
IL	LD　16#45 ROL　2 ST　Var1　（*结果 Var1 为 16#15*）
FBD	ROL 2#1001_0011 — 2 — Var1=2#01001110

3.4.4 ROR——循环右移指令

- 功能：对操作数进行按位循环右移，右边移出的位直接补充到左边最高位。
- 输入输出数据类型：BYTE、INT、WORD、DWORD 等。
  - ✧ 指令使用举例

    变量声明：Var1: BYTE;

    Var2: INT;

POU 语言	程　序
LD	ROR EN 2#1001_0011 2 Var1=2#11100100
ST	Var1:=ROR(16#45,2)　（*结果 Var1 为 16#51*） Var2:= ROR (16#45,2)　（*结果 Var2 为 16#4011*） 注意：在循环右移过程中，虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
IL	LD　16#45 ROR　2 ST　Var1　（*结果 Var1 为 16#51*）
FBD	ROR 2#1001_0011 2 Var1=2#11100100

3.5 选择运算指令

所有的选择运算指令在执行时均可以带有变量。为了能够更加清楚地说明问题，以下各例只使用常量。

3.5.1 SEL——二选一指令

- 功能：通过选择开关在两个输入数据中选择一个作为输出，选择开关为 FALSE 时输出为第一个输入数据，选择开关为 TRUE 时输出为第二个输入数据。
- 指令格式：SEL(G,IN0,IN1)，其中 G 为选择开关，IN0 和 IN1 分别为第一个输入数据和第二个输入数据。
- 输入输出数据类型：G 必须是 BOOL 类型，IN0、IN1 和输出数据可以是任意数据类型。
  - ✧ 指令使用举例

    变量声明：Var1: INT;

    Var2: INT;

POU 语言	程　序
LD	SEL EN TRUE 2 6 Var1=6

ST	Var1:=SEL(TRUE,3,4); (*结果 Var1 为 4 *)
IL	LD TRUE SEL 2,6 ST Var1 (*结果 Var1 为 6*) LD FALSE SEL 2,6 ST Var2 (*结果 Var2 为 2*)
FBD	SEL: TRUE, 2, 6 —Var1=6

3.5.2 MAX——取最大值指令

- 功能：在多个输入数据中选择最大值作为输出。
- 指令格式：OUT=MAX(IN1,…,INn)，其中 IN1 和 INn 分别为第 1 个输入数据和第 n 个输入数据，OUT 是输出数据。
- 输入输出数据类型：IN1,…,INn 和 OUT 可以是任意数据类型。
  - ✧ 指令使用举例

    变量声明：Var1: INT;
    Var2: INT;

POU 语言	程　序
LD	MAX: EN, 90, 60 —Var1=90
ST	Var1:=MAX(90,60); (*结果 Var1 为 90 *) Var2:=MAX(40,MAX(90,60)); (*结果 Var2 为 90 *)
IL	LD 90 MAX 40 MAX 70 MAX 60 ST Var1 （*结果 Var1 为 90*）
FBD	MAX(90, 40) — MAX(70) — MAX(60) —Var1=90

3.5.3 MIN——取最小值指令

- 功能：在多个输入数据中选择最小值作为输出。
- 指令格式：OUT=MIN(IN1,…,INn)，其中 IN1 和 INn 分别为第 1 个输入数据和第 n 个输入数据，OUT 是输出数据。
- 输入输出数据类型：IN1,…,INn 和 OUT 可以是任意数据类型。
  - ✧ 指令使用举例

    变量声明：Var1: INT;
    Var2: INT;

POU 语言	程　序
LD	MIN EN 90 60 Var1=60
ST	Var1:=MIN(90,30); (*结果 Var1 为 30 *); Var2:=MIN(MIN(90,30),60); (*结果 Var2 为 30 *);
IL	LD 90 MIN 30 MIN 40 MIN 70 ST Var1 （*结果 Var1 为 30*）
FBD	MIN 90 30 — MIN 70 — MIN 60 — Var1=30

3.5.4 LIMIT——极限值指令

- **功能**：判断输入数据是否在最小值和最大值之间，若输入数据在二者之间，则直接把输入数据作为输出数据进行输出。若输入数据大于最大值，则把最大值作为输出值。若输入数据小于最小值，则把最小值作为输出值。
- **指令格式**：OUT=LIMIT(Min,IN,Max)
- **输入输出数据类型**：IN 和 OUT 可以是任意数据类型。
  - 指令使用举例

    变量声明：Var1: INT;

POU 语言	程　序
LD	LIMIT EN 5 11 10 Var1=10 （*11 为输入数据，5 为最小值，10 为最大值*）
ST	Var1:=LIMIT(30,90,80); (*结果 Var1 为 80 *)
IL	LD 90 LIMIT 30，80 ST Var1 (*结果 Var1 为 80*)
FBD	LIMIT 5 11 10 Var1=10 （*11 为输入数据，5 为最小值，10 为最大值*）

3.5.5 MUX——多选一指令

- **功能**：通过控制数在多个输入数据中选择一个作为输出。
- **指令格式**： OUT:=MUX(K,IN0,…,INn)，其中 K 为控制数，IN0,…,INn 为输入数据，OUT 为输出结果。控制数为 K 时选择第 IN_k 个输入数据作为输出。
- **输入输出数据类型**：IN0,…, INn 和 OUT 可以是任意数据类型，K 必须是 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 或 UDINT。

✧ 指令使用举例

变量声明：Var1: INT;

POU 语言	程 序
LD	MUX EN; 2, 10, 20, 30, 40 → Var1=30 （*2 为控制数据，对应于 30，所以输出 30 *）
ST	Var1:=MUX(0,30,40,50,60,70,80);（*结果 Var1 为 30*）
IL	LD 0 MUX 30，40，50，60，70，80 ST Var1 （*结果 Var1 为 30*）
FBD	MUX; 2, 10, 20, 30, 40 → Var1=30 （*2 为控制数据，对应于 30，所以输出 30*）

3.6 比较运算指令

3.6.1 GT——大于指令

- **功能**：判断两个操作数的大小，当第一个数大于第二个数时输出 TRUE，否则输出为 FALSE。
- **输入输出数据类型**：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。

✧ 指令使用举例

变量声明：Var1: BOOL;

POU 语言	程 序
LD	GT EN; 20, 30 → Var1 （*结果 Var1 为 FALSE*）

ST	Var1:=20>30;
IL	LD 20 GT 30 ST Var1 （*结果 Var1 为 FALSE*）
FBD	GT 20 30 Var1 （*结果 Var1 为 FALSE*）

3.6.2 LT——小于指令

◆ 功能：判断两个操作数的大小，当第一个数小于第二个数时返回 TRUE，否则结果为 FALSE。

◆ 输入输出数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。

✧ 指令使用举例

变量声明：Var1: BOOL;

POU 语言	程 序
LD	LT EN 20 30 Var1 （*结果 Var1 为 TRUE*）
ST	VAR1:=20<30; （*结果 Var1 为 TRUE*）
IL	LD 20 LT 30 ST Var1 (*结果 Var1 为 TRUE*)
FBD	LT 20 30 Var1 （*结果 Var1 为 TRUE*）

3.6.3 GE——大于等于指令

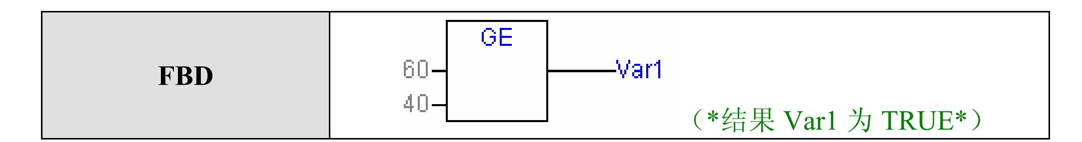
◆ 功能：判断两个操作数的大小，当第一个数大于等于第二个数时返回 TRUE，否则结果为 FALSE。

◆ 输入输出数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。

✧ 指令使用举例

变量声明：Var1: BOOL;

POU 语言	程 序
LD	GE EN 60 40 Var1 （*结果 Var1 为 TRUE*）
ST	VAR1:=60>=40; （*结果 Var1 为 TRUE*）
IL	LD 60 GE 40 ST Var1 （*结果 Var1 为 TRUE*）



FBD	GE: 60, 40 → Var1　（*结果 Var1 为 TRUE*）

3.6.4 LE——小于等于指令

◆ 功能：判断两个操作数的大小，当第一个数小于等于第二个数时返回 TRUE，否则结果为 FALSE。

◆ 输入输出数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。

✧ 指令使用举例

变量声明：Var1: BOOL;

POU 语言	程　序
LD	LE: EN, 20, 30 → Var1　（*结果 Var1 为 TRUE*）
ST	VAR1:=20<=30;　（*结果 Var1 为 TRUE*）
IL	LD 20 LE 30 ST Var1　（*结果 Var1 为 TRUE*）
FBD	LE: 20, 30 → Var1　（*结果 Var1 为 TRUE*）

3.6.5 EQ——等于指令

◆ 功能：判断两个操作数是否相等，当第一个数等于第二个数时返回 TRUE，否则结果为 FALSE。

◆ 输入输出数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。

✧ 指令使用举例

变量声明：Var1: BOOL;

POU 语言	程　序
LD	EQ: EN, 40, 40 → Var1　（*结果 Var1 为 TRUE*）
ST	VAR1:=40=40;　（*结果 Var1 为 TRUE*）
IL	LD 40 EQ 40 ST Var1　（*结果 Var1 为 TRUE*）
FBD	EQ: 40, 40 → Var1　（*结果 Var1 为 TRUE*）

3.6.6 NE——不等于指令

- **功能：**判断两个操作数是否不相等，当第一个数不等于第二个数时返回 TRUE，否则结果为 FALSE。
- **输入输出数据类型：**BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TIME_OF_DAY、DATE_AND_TIME 和 STRING。
  - ✧ **指令使用举例**

    变量声明：Var1: BOOL;

POU 语言	程　序
LD	NE EN 40 — 40 — ——Var1 （*结果 Var1 为 FALSE*）
ST	VAR1:=40<>40;　（*结果 Var1 为 FALSE*）
IL	LD　40 NE　40 ST　Var1　（*结果 Var1 为 FALSE*）
FBD	NE 40 — 40 — ——Var1 （*结果 Var1 为 FALSE*）

3.7　数据类型转换指令

PowerPro 提供了共计 240 个数据类型转换指令，用于各种数据类型之间相互转换。

- 语法：<TYPE1>_TO_<TYPE2>

注意

- ➢ <TYPE>_TO_STRING 的转换中，字符串是从左边开始生成的。如果定义的字符串长度小于<TYPE>的长度，右边部分会被截去。
- ➢ 禁止将“较大的”数据类型隐含地转换为“较小的”数据类型使用，例如将 INT 类型转换为 BYTE 类型使用，或者将 DINT 类型转换为 WORD 类型。如果需要进行数据类型转换使用，则需要使用下表提供的类型装换指令进行转换之后再使用。

- **数据类型转换指令列表：**表 3-7-1 列出了所有的数据类型转换指令，本小节只讲述主要的数据类型转换指令。

表 3-7-1

BOOL_TO_<TYPE>	BYTE_TO_<TYPE>	DATE_TO_<TYPE>	DINT_TO_<TYPE>
BOOL_TO_BYTE	BYTE_TO_BOOL	DATE_TO_BOOL	DINT_TO_BOOL
BOOL_TO_DATE	BYTE_TO_DATE	DATE_TO_BYTE	DINT_TO_BYTE
BOOL_TO_DINT	BYTE_TO_DINT	DATE_TO_DINT	DINT_TO_DATE
BOOL_TO_DT	BYTE_TO_DT	DATE_TO_DT	DINT_TO_DT
BOOL_TO_DWORD	BYTE_TO_DWORD	DATE_TO_DWORD	DINT_TO_DWORD
BOOL_TO_INT	BYTE_TO_INT	DATE_TO_INT	DINT_TO_INT
BOOL_TO_REAL	BYTE_TO_REAL	DATE_TO_REAL	DINT_TO_REAL
BOOL_TO_SINT	BYTE_TO_SINT	DATE_TO_SINT	DINT_TO_SINT
BOOL_TO_STRING	BYTE_TO_STRING	DATE_TO_STRING	DINT_TO_STRING
BOOL_TO_TIME	BYTE_TO_TIME	DATE_TO_TIME	DINT_TO_TIME
BOOL_TO_TOD	BYTE_TO_TOD	DATE_TO_TOD	DINT_TO_TOD
BOOL_TO_UDINT	BYTE_TO_UDINT	DATE_TO_UDINT	DINT_TO_UDINT
BOOL_TO_UINT	BYTE_TO_UINT	DATE_TO_UINT	DINT_TO_UINT
BOOL_TO_USINT	BYTE_TO_USINT	DATE_TO_USINT	DINT_TO_USINT
BOOL_TO_WORD	BYTE_TO_WORD	DATE_TO_WORD	DINT_TO_WORD

DT_TO_<TYPE>	DWORD_TO_<TYPE>	INT_TO_<TYPE>	WORD_TO_<TYPE>
DT_TO_BOOL	DWORD_TO_BOOL	INT_TO_BOOL	WORD_TO_BOOL
DT_TO_BYTE	DWORD_TO_BYTE	INT_TO_BYTE	WORD_TO_BYTE
DT_TO_DATE	DWORD_TO_DATE	INT_TO_DATE	WORD_TO_DATE
DT_TO_DINT	DWORD_TO_DINT	INT_TO_DINT	WORD_TO_DINT
DT_TO_DWORD	DWORD_TO_DT	INT_TO_DT	WORD_TO_DT
DT_TO_INT	DWORD_TO_INT	INT_TO_DWORD	WORD_TO_DWORD
DT_TO_REAL	DWORD_TO_REAL	INT_TO_REAL	WORD_TO_INT
DT_TO_SINT	DWORD_TO_SINT	INT_TO_SINT	WORD_TO_REAL
DT_TO_STRING	DWORD_TO_STRING	INT_TO_STRING	WORD_TO_SINT
DT_TO_TIME	DWORD_TO_TIME	INT_TO_TIME	WORD_TO_STRING
DT_TO_TOD	DWORD_TO_TOD	INT_TO_TOD	WORD_TO_TIME
DT_TO_UDINT	DWORD_TO_UDINT	INT_TO_UDINT	WORD_TO_TOD
DT_TO_UINT	DWORD_TO_UINT	INT_TO_UINT	WORD_TO_UDINT
DT_TO_USINT	DWORD_TO_USINT	INT_TO_USINT	WORD_TO_UINT
DT_TO_WORD	DWORD_TO_WORD	INT_TO_WORD	WORD_TO_USINT
REAL_TO_<TYPE>	**SINT_TO_<TYPE>**	**STRING_TO_<TYPE>**	**TIME_TO_<TYPE>**
REAL_TO_BOOL	SINT_TO_BOOL	STRING_TO_BOOL	TIME_TO_BOOL
REAL_TO_BYTE	SINT_TO_BYTE	STRING_TO_BYTE	TIME_TO_BYTE
REAL_TO_DATE	SINT_TO_DATE	STRING_TO_DATE	TIME_TO_DATE
REAL_TO_DINT	SINT_TO_DINT	STRING_TO_DINT	TIME_TO_DINT
REAL_TO_DT	SINT_TO_DT	STRING_TO_DT	TIME_TO_DT
REAL_TO_DWORD	SINT_TO_DWORD	STRING_TO_DWORD	TIME_TO_DWORD
REAL_TO_INT	SINT_TO_INT	STRING_TO_INT	TIME_TO_INT
REAL_TO_SINT	SINT_TO_REAL	STRING_TO_REAL	TIME_TO_REAL
REAL_TO_STRING	SINT_TO_STRING	STRING_TO_SINT	TIME_TO_SINT
REAL_TO_TIME	SINT_TO_TIME	STRING_TO_TIME	TIME_TO_STRING
REAL_TO_TOD	SINT_TO_TOD	STRING_TO_TOD	TIME_TO_TOD
REAL_TO_UDINT	SINT_TO_UDINT	STRING_TO_UDINT	TIME_TO_UDINT
REAL_TO_UINT	SINT_TO_UINT	STRING_TO_UINT	TIME_TO_UINT
REAL_TO_USINT	SINT_TO_USINT	STRING_TO_USINT	TIME_TO_USINT
REAL_TO_WORD	SINT_TO_WORD	STRING_TO_WORD	TIME_TO_WORD
TOD_TO_<TYPE>	**UDINT_TO_<TYPE>**	**UINT_TO_<TYPE>**	**USINT_TO_<TYPE>**
TOD_TO_BOOL	UDINT_TO_BOOL	UINT_TO_BOOL	USINT_TO_BOOL
TOD_TO_BYTE	UDINT_TO_BYTE	UINT_TO_BYTE	USINT_TO_BYTE
TOD_TO_DATE	UDINT_TO_DATE	UINT_TO_DATE	USINT_TO_DATE
TOD_TO_DINT	UDINT_TO_DINT	UINT_TO_DINT	USINT_TO_DINT
TOD_TO_DT	UDINT_TO_DT	UINT_TO_DT	USINT_TO_DT
TOD_TO_DWORD	UDINT_TO_DWORD	UINT_TO_DWORD	USINT_TO_DWORD
TOD_TO_INT	UDINT_TO_INT	UINT_TO_INT	USINT_TO_INT
TOD_TO_REAL	UDINT_TO_REAL	UINT_TO_REAL	USINT_TO_REAL
TOD_TO_SINT	UDINT_TO_SINT	UINT_TO_SINT	USINT_TO_SINT
TOD_TO_STRING	UDINT_TO_STRING	UINT_TO_STRING	USINT_TO_STRING
TOD_TO_TIME	UDINT_TO_TIME	UINT_TO_TIME	USINT_TO_TIME
TOD_TO_UDINT	UDINT_TO_TOD	UINT_TO_TOD	USINT_TO_TOD
TOD_TO_UINT	UDINT_TO_UINT	UINT_TO_UDINT	USINT_TO_UDINT
TOD_TO_USINT	UDINT_TO_USINT	UINT_TO_USINT	USINT_TO_UINT
TOD_TO_WORD	UDINT_TO_WORD	UINT_TO_WORD	USINT_TO_WORD

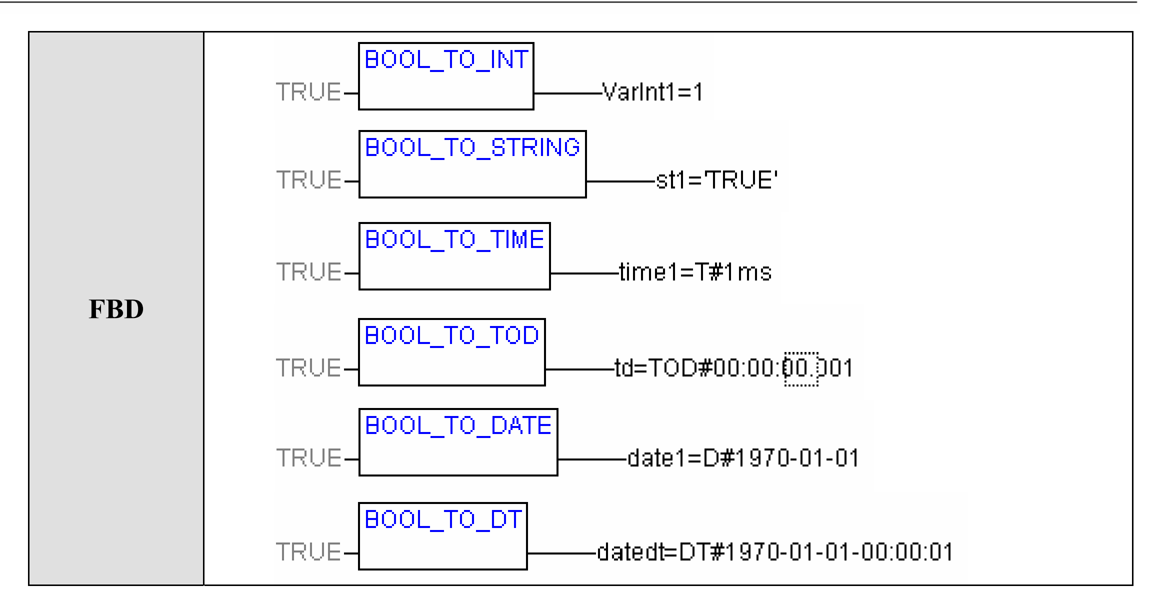
3.7.1 BOOL_TO_<TYPE>——布尔类型转换指令

- ◆ 功能：把布尔数据类型转换为其它数据类型。
- ◆ 输入输出数据类型：
  - ✓ 输出为数字类型时，如果输入是 TRUE，则输出 1，如果输入是 FALSE，则输出为 0。
  - ✓ 输出为字符串类型时，如果输入是 TRUE，则输出字符串'TRUE'，如果输入是 FALSE，则输出为字符串'FALSE'。
  - ✧ 指令使用举例

    变量声明：

    VarInt1: INT;
    st1: STRING;
    time1: TIME;
    td: TOD;
    date1: DATE;
    datedt: DT;

<table>
<tr><th>POU 语言</th><th>程　序（部 分）</th></tr>
<tr><td>LD</td><td>BOOL_TO_INT: EN, TRUE — VarInt1=1
BOOL_TO_STRING: EN, TRUE — st1='TRUE'
BOOL_TO_TIME: EN, TRUE — time1=T#1ms
BOOL_TO_TOD: EN, TRUE — td=TOD#00:00:00.001
BOOL_TO_DATE: EN, TRUE — date1=D#1970-01-01
BOOL_TO_DT: EN, TRUE — datedt=DT#1970-01-01-00:00:01</td></tr>
<tr><td>ST</td><td>VarInt1:=BOOL_TO_INT(TRUE); (*结果为 1*)
st1:=BOOL_TO_STRING(TRUE); (*结果为’TRUE’*)
time1:=BOOL_TO_TIME(TRUE); (*结果为 T#1ms*)
td:=BOOL_TO_TOD(TRUE); (*结果为 TOD#00:00:00.001*)
date1:=BOOL_TO_DATE(TRUE); (*结果为 D#1970-01-01*)
datedt :=BOOL_TO_DT(TRUE); (*结果为 DT#1970-01-01-00:00:01*)</td></tr>
<tr><td rowspan="6">IL</td><td>LD TRUE
BOOL_TO_INT
ST VarInt1 (*结果为 1*)</td></tr>
<tr><td>LD TRUE
BOOL_TO_STRING
ST st1 (*结果为’TRUE’*)</td></tr>
<tr><td>LD TRUE
BOOL_TO_TIME
ST time1 (*结果为 T#1ms*)</td></tr>
<tr><td>LD TRUE
BOOL_TO_TOD
ST td (*结果为 TOD#00:00:00.001*)</td></tr>
<tr><td>LD TRUE
BOOL_TO_DATE
ST date1 (*结果为 D#1970-01-01*)</td></tr>
<tr><td>LD TRUE
BOOL_TO_DT
ST datedt (*结果为 DT#1970-01-01-00:00:01*)</td></tr>
</table>



3.7.2 <TYPE>_TO_BOOL——布尔类型生成指令

◆ 功能：把其它数据类型转换为布尔数据类型。

◆ **输入输出数据类型：**

✓ 输入为数字类型时，当输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

✓ 输入字符串类型时，当输入是'TRUE'时输出为 TRUE，否则输出为 FALSE。

✧ **指令使用举例**

变量声明：

VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL3: BOOL;
VarBOOL4: BOOL;

POU 语言	程 序（部 分）
LD	BYTE_TO_BOOL: EN, 213 → VarBOOL1 (*结果为 TRUE *) INT_TO_BOOL: EN, 0 → VarBOOL2 (*结果为 FALSE *) TIME_TO_BOOL: EN, T#5ms → VarBOOL3 (*结果为 TRUE *) STRING_TO_BOOL: EN, 'TRUE' → VarBOOL4 (*结果为 TRUE *)
ST	VarBOOL1:= BYTE_TO_BOOL(2#11010101); (*结果为 TRUE *) VarBOOL2:= INT_TO_BOOL(0); (*结果为 FALSE *) VarBOOL3:= TIME_TO_BOOL(T#5ms); (*结果为 TRUE *) VarBOOL4:= STRING_TO_BOOL('TRUE'); (*结果为 TRUE *)

IL	LD 213 BYTE_TO_BOOL ST VarBOOL1 (*结果为 TRUE*)
	LD 0 INT_TO_BOOL ST VarBOOL2 (*结果为 FALSE*)
	LD T#5ms TIME_TO_BOOL ST VarBOOL3 (*结果为 TRUE*)
	LD 'TRUE' STRING_TO_BOOL ST VarBOOL4 (*结果为 TRUE*)
FBD	2#11010101 — BYTE_TO_BOOL — VarBOOL1 (*结果为 TRUE*) 0 — INT_TO_BOOL — VarBOOL2 (*结果为 FALSE*) T#5ms — TIME_TO_BOOL — VarBOOL3 (*结果为 TRUE*) 'TRUE' — STRING_TO_BOOL — VarBOOL4 (*结果为 TRUE*)

3.7.3 INT_TO_<TYPE>——整数类型转换指令

◆ 功能：把整型数据类型转换为其它数据类型。

✧ 指令使用举例

变量声明：

VarSINT1: SINT;

VarREAL1: REAL;

POU 语言	程 序（部 分）
LD	INT_TO_SINT EN 4223 — INT_TO_SINT — VarSINT1=127
ST	VarSINT1 := INT_TO_SINT(4223); (*结果 VarSINT1 为 127 *) 说明：如果将整数 4223（十六进制为 16#107F）保存为 SINT 型变量，则会丢失高位数据，只显示低位数据 127（十六进制为 16#7F）。
IL	LD 2 INT_TO_REAL ST VarREAL1 (*结果 VarREAL1 为 2.0*)
FBD	4223 — INT_TO_SINT — VarSINT1=127

注意

➢ 当从较大类型数据转为较小类型数据时，有可能丢失信息。如果要转换的值超过范围限制，则这个数的高字节被忽略。

3.7.4 REAL_TO_<TYPE>——实数类型转换指令

◆ 功能：把浮点数转换为其它类型数据。把浮点数转换为其它类型数据时，先将值近似成最接近的整数值，然后转成新的变量类型。

✧ 指令使用举例

变量声明：

VarINT1: INT;
VarINT2: INT;
VarINT3: INT;
VarINT4: INT;

POU 语言	程　序（部 分）
LD	REAL_TO_INT EN 1.5 — VarINT1=2
ST	VarINT1:= REAL_TO_INT(1.5);　(*结果 VarINT1 为 2*) VarINT2:= REAL_TO_INT(1.4);　(*结果 VarINT2 为 1*) VarINT3:= REAL_TO_INT(-1.5);　(*结果 VarINT3 为-2*) VarINT4:= REAL_TO_INT(-1.4);　(*结果 VarINT4 为-1*)
IL	LD　2.7 REAL_TO_INT ST　VarINT1　(*结果 VarINT1 为 3*)
FBD	REAL_TO_INT 1.5 — VarINT1=2

注意

➢ 当从较大类型数据转为较小类型数据时，有可能丢失信息。

3.7.5 TIME_TO_<TYPE>——时间类型转换指令

◆ 功能：把时间型数据转换为其它类型数据，时间在内部以毫秒为单位存储成 DWORD 类型（对于 TIME_OF_DAY 变量从凌晨 00：00 开始）。当从较大类型转为较小类型时，有可能丢失信息。

✧ 指令使用举例

变量声明：

Varstr: STRING;
Vardword: DWORD;

POU 语言	程　序（部 分）
LD	TIME_TO_STRING EN T#12ms — Varstr='T#12ms' TIME_TO_DWORD EN T#5m — Vardword=300000
ST	Varstr:=TIME_TO_STRING(T#12ms);　(*结果为‘T#12ms’*) Vardword:=TIME_TO_DWORD(T#5m);　(*结果为 300000*)

<table>
<tr><td rowspan="2">IL</td><td>LD T#12ms
TIME_TO_STRING
ST Varstr (*结果为‘T#12ms’*)</td></tr>
<tr><td>LD T#300000ms
TIME_TO_DWORD
ST Vardword (*结果为 300000*)</td></tr>
<tr><td>FBD</td><td>T#12ms — TIME_TO_STRING — Varstr='T#12ms'
T#300000ms — TIME_TO_DWORD — Vardword=300000</td></tr>
</table>

3.7.6 DATE_TO_<TYPE>——日期类型转换指令

◆ 功能：把日期型数据转换为其它类型数据，日期在内部以秒为单位存储成 DWORD，时间从 1970 年 1 月 1 日开始。

✧ 指令使用举例

变量声明：

VarInt1: INT;

VarStr1: STRING;

<table>
<tr><th>POU 语言</th><th>程 序（部 分）</th></tr>
<tr><td>LD</td><td>DATE_TO_STRING: EN; D#1970-01-01 — VarStr1='D#1970-01-01'
DATE_TO_INT: EN; D#1970-01-15 — VarInt1=29952</td></tr>
<tr><td>ST</td><td>VarStr1:=DATE_TO_STRING(D#1970-01-01);
(*结果为'D#1970-01-01' *)
VarInt1:=DATE_TO_INT(D#1970-01-15); (*结果为 29952*)</td></tr>
<tr><td rowspan="2">IL</td><td>LD D#1970-01-01
DATE_TO_STRING
ST VarStr1 (*结果为“TRUE”*)</td></tr>
<tr><td>LD D#1970-01-15
DATE_TO_INT
ST VarInt1 (*结果为 29952*)</td></tr>
<tr><td>FBD</td><td>D#1970-01-01 — DATE_TO_STRING — VarStr1='D#1970-01-01'
D#1970-01-15 — DATE_TO_INT — VarInt1=29952</td></tr>
</table>

3.7.7 DT_TO_<TYPE>——日期时间类型转换指令

◆ 功能：把日期时间型数据转换为其它类型数据，日期在内部以秒为单位存储成 DWORD，时间从 1970 年 1 月 1 日开始。

✧ 指令使用举例

变量声明：

Varbyte: BYTE;
Varstr: STRING;

POU 语言	程　序（部 分）
LD	DT_TO_BYTE: EN; DT#1970-01-15-05:05:05 → Varbyte=129 DT_TO_STRING: EN; DT#1998-02-13-14:20 → Varstr='DT#1998-02-13-14:20:00
ST	Varbyte:=DT_TO_BYTE(DT#1970-01-15-05:05:05); (*结果 Varbyte 为 129*) Varstr:=DT_TO_STRING(DT#1998-02-13-14:20); (*结果 Varstr 为'DT#1998-02-13-14:20')
IL	LD DT#1970-01-15-05:05:05 DT_TO_BYTE ST Varbyte (*结果 Varbyte 为 129*) LD DT#1998-02-13-14:20 DT_TO_STRING ST Varstr (*结果 Varstr 为'DT#1998-02-13-14:20')
FBD	DT_TO_BYTE: DT#1970-01-15-05:05:05 → Varbyte=129 DT_TO_STRING: DT#1998-02-13-14:20 → Varstr='DT#1998-02-13-14:20:00

3.7.8 STRING_TO_<TYPE>——字符类型转换指令

◆ 功能：把字符串转换为其它类型数据，字符串型变量操作数必须包含一个有效的目标变量值，否则转换结果为 0。

✧ 指令使用举例

变量声明：

Varword: WORD;
Vartime: TIME;

POU 语言	程　序（部 分）
LD	STRING_TO_WORD: EN; 'Hollysys' → Varword=0 STRING_TO_TIME: EN; 'T#127ms' → Vartime=T#127ms
ST	Varword:=STRING_TO_WORD('Hollysys'); (*结果为 0*) Vartime:=STRING_TO_TIME('T#127ms'); (*结果为 T#127ms*)

IL	LD 'Hollysys' STRING_TO_WORD ST Varword (*结果 Varword 为 0*)
	LD 'T#127ms' STRING_TO_TIME ST Vartime (*结果 Vartime 为 T#127ms*)
FBD	'Hollysys' — STRING_TO_WORD — Varword=0 'T#127ms' — STRING_TO_TIME — Vartime=T#127ms

3.7.9 TRUNC——截短转换指令

◆ 功能：REAL 型转换为 INT 型，该值的小数部分被截取，只保留整数部分。

✧ 指令使用举例

变量声明：

Varint1: INT;

Varint2: INT;

POU 语言	程　序
LD	TRUNC EN 1.9 — Varint1=1 TRUNC EN -1.4 — Varint2=-1
ST	Varint1:=TRUNC(1.9); (*结果 Varint1 为 1*) Varint2:=TRUNC(-1.4); (*结果 Varint2 为-1*)
IL	LD 1.9 TRUNC ST Varint1 (*结果 Varint1 为 1*)
	LD -1.4 TRUNC ST Varint2 (*结果 Varint2 为-1*)
FBD	1.9 — TRUNC — Varint1=1 -1.4 — TRUNC — Varint2=-1

注意

➢ 当从较大类型数据转为较小类型数据时，有可能丢失信息。

3.8 初等数学运算指令

3.8.1 ABS——绝对值指令

◆ 功能：把输入数据的绝对值赋予输出变量。
输入数据类型和输出数据类型的组合见下表。

输入数据类型	输出数据类型
INT	INT、WORD、DWORD、DINT、UINT、REAL
REAL	REAL
BYTE	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
WORD	WORD、DWORD、DINT、REAL
DWORD	DWORD、DINT、REAL
SINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
USINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL
UINT	WORD、DWORD、DINT、UINT、REAL
DINT	DWORD、DINT、REAL
UDINT	DWORD、DINT、UDINT、REAL

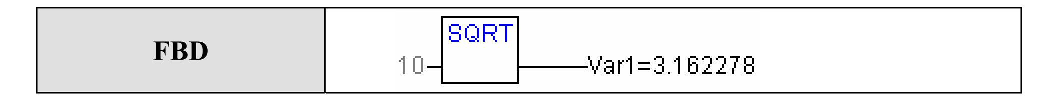
✧ 指令使用举例
变量声明：
Varint1: INT;

POU 语言	程　序
LD	ABS EN -2 → Varint1=2
ST	i:=ABS(-2);　(*结果 Varint1 为 2*)
IL	LD　-2 ABS ST　Varint1　(*结果 Varint1 为 2*)
FBD	ABS -2 → Varint1=2

3.8.2 SQRT——平方根指令

◆ 功能：对输入数据求平方根，输入数据为非负数。
◆ **输入输出数据类型**：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 类型。
✧ 指令使用举例
变量声明：
Var1: REAL;

POU 语言	程　序
LD	SQRT EN 10 → Var1=3.162278
ST	Var1:=SQRT(10); (*结果 Var1 为 3.162278*)
IL	LD　10 SQRT ST　Var1　(*结果 Var1 为 3.162278*)

FBD	

3.8.3 LN——自然对数指令

- 功能：对输入数据求自然对数，输入数据必须为正数。
- 输入输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。
  - 指令使用举例
    变量声明：
        Var1: REAL;

POU 语言	程　序
LD	LN EN 45 Var1=3.806663
ST	Var1:=LN(45);　(*结果 Var1 为 3.806663*)
IL	LD　45 LN ST　Var1　(*结果 Var1 为 3.806663*)
FBD	LN 45 Var1=3.806663

3.8.4 LOG——常用对数指令

- 功能：对输入数据求以 10 为底的对数，输入数据必须为正数。
- 输入输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。
  - 指令使用举例
    变量声明：
        Var1: REAL;

POU 语言	程　序
LD	LOG EN 314.5 Var1=2.497621
ST	Var1:=LOG(314.5);　(*结果 Var1 为 2.497621 *)
IL	LD　314.5 LOG ST　Var1　(*结果 Var1 为 2.497621 *)
FBD	LOG 314.5 Var1=2.497621

3.8.5 EXP——指数指令

- 功能：对输入数据求指数，即 $y = e^x$，其中 X 为输入，y 为输出。
- 输入输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程 序
LD	EXP / EN / 2 / Var1=7.389056
ST	Var1:=EXP(2); （*结果 Var1 为 7.389056*）
IL	LD 2 EXP ST Var1 （*结果 Var1 为 7.389056*）
FBD	EXP / 2 / Var1=7.389056

3.8.6 SIN——正弦指令

◆ **功能**：求输入数据的正弦值，输入数据以弧度表示。

✓ $\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^{\circ}}$

◆ **输入输出数据类型**：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程 序
LD	SIN / EN / 1.5 / Var1=0.997495
ST	Var1:=SIN(1.5); (*结果 Var1 为 0.997495 *)
IL	LD 1.5 SIN ST Var1 (*结果 Var1 为 0.997495 *)
FBD	SIN / 1.5 / Var1=0.997495

3.8.7 COS——余弦指令

◆ **功能**：求输入数据的余弦值，输入数据以弧度表示。

✓ $\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^{\circ}}$

◆ **输入输出数据类型**：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程　序
LD	COS EN 0.5 Var1=0.8775826
ST	Var1:=COS(0.5); (*结果 Var1 为 0.8775826 *)
IL	LD 0.5 COS ST Var1 (*结果 Var1 为 0.8775826 *)
FBD	COS 0.5 Var1=0.8775826

3.8.8 TAN——正切指令

◆ 功能：求输入数据的正切值，输入数据以弧度表示。

✓ $弧度(rad) = 角度 * \frac{\pi}{180^{\circ}}$

◆ **输入输出数据类型**：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程　序
LD	TAN EN 0.5 Var1=0.5463025
ST	Var1:=TAN(0.5); (*结果 Var1 为 0.5463025 *)
IL	LD 0.5 TAN ST Var1 (*结果 Var1 为 0.5463025 *)
FBD	TAN 0.5 Var1=0.5463025

3.8.9 ASIN——反正弦指令

◆ 功能：求输入数据的反正弦值。

✓ $弧度(rad) = 角度 * \frac{\pi}{180^{\circ}}$

◆ **输入输出数据类型**：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程　序
LD	ASIN EN 0.5 — Var1=0.5235988
ST	Var1:=ASIN(0.5);　(*结果 Var1 为 0.5235988 *)
IL	LD　0.5 ASIN ST　Var1　(*结果 Var1 为 0.5235988 *)
FBD	ASIN 0.5 — Var1=0.5235988

3.8.10　ACOS——反余弦指令

◆ 功能：求输入数据的反余弦值。

✓ $弧度(rad) = 角度 * \frac{\pi}{180^\circ}$

◆ **输入输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程　序
LD	ACOS EN 0.5 — Var1=1.047198
ST	Var1:=ACOS(0.5);　(*结果 Var1 为 1.047198 *)
IL	LD　0.5 ACOS ST　Var1　(*结果 Var1 为 1.047198 *)
FBD	ACOS 0.5 — Var1=1.047198

3.8.11　ATAN——反正切指令

◆ 功能：求输入数据的反正切值。

✓ $弧度(rad) = 角度 * \frac{\pi}{180^\circ}$

◆ **输入输出数据类型：**输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

✧ 指令使用举例

变量声明：

Var1: REAL;

POU 语言	程　序
LD	ATAN EN 0.5 Var1=0.4636476
ST	Var1:=ATAN(0.5); (*结果 Var1 为 0.4636476 *)
IL	LD 0.5 ATAN ST Var1 (*结果 Var1 为 0.4636476 *)
FBD	ATAN 0.5 Var1=0.4636476

3.8.12 EXPT——幂指令

- **功能：**对输入数据求幂，输入数据 1 为幂底数，输入数据 2 为幂指数。
- **输入输出数据类型：**输入数据的类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、UINT、UDINT，输出数据必须是 REAL 型。
  - **指令使用举例**

    变量声明：

    Var1: REAL;

POU 语言	程　序
LD	EXPT EN 7 2 Var1=49
ST	Var1:=EXPT(7,2); (*结果 Var1 为 49 *)
IL	LD 7 EXPT 2 ST Var1 (*结果 Var1 为 49 *)
FBD	EXPT 7 2 Var1=49

3.9 地址运算指令

3.9.1 ADR——取地址指令

- **功能：**取得输入变量的内存地址，并输出。该地址可以在程序内当作指针使用，也可以作为指针传送给函数。
  - **指令使用举例**

    变量声明：

    Var1: BYTE;

    VarAddress: POINTER TO BYTE;

POU 语言	程　序
LD	ADR EN Var1=0 VarAddress=<01789890>

ST	VarAddress:= ADR(Var1);
IL	LD Var1 ADR ST VarAddress
FBD	ADR Var1=0 — VarAddress=<0178b240>

3.9.2 ^——取地址内容指令

◆ 功能：在指针变量后增加“^”符号，以取得该指针所指地址的数据。

✧ 指令使用举例

变量声明：

VarAddress: POINTER TO BYTE;
Var1: BYTE;
Var2: BYTE;

POU 语言	程　序
LD	ADR EN Var1=100 — VarAddress=<0178bc84> MOVE EN VarAddress^=100 — Var2=100
ST	VarAddress:= ADR(Var1); Var2:= VarAddress^; (*结果 Var2 为 100 *)
FBD	ADR Var1=100 — VarAddress=<0178df84> MOVE VarAddress^=100 — Var2=100

3.10 调用运算指令

3.10.1 CAL——调用运算指令

◆ 功能：调用功能块或者程序。在 IL 语言中使用 CAL 运算来调用功能块或者程序。被调用功能块/程序的输入变量位于该功能块/程序名称右侧的括号内。

✧ 指令使用举例

在程序中调用名称为 Inst 的功能块，其输入变量分别为 Par1=0, Par2=TRUE。

IL 中调用如下：

CAL Inst(Par1:=0, Par2:=TRUE)

第4章 PowerPro G3 指令

4.1 字符串处理指令（Standard.lib）

4.1.1 LEN——取字符串长度指令

- 功能：计算字符串的长度。
- 输入输出数据类型：输入是 STRING 类型，输出是 INT 类型。

✧ 指令使用举例

变量声明：

Var1INT1: INT;

POU 语言	程 序
LD	LEN: EN; 'Hollysys'—STR; —VarINT1=8
ST	Var1INT1 := LEN (‘Hollysys’); (*结果 Var1INT1 为 8*)
IL	LD ‘Hollysys’ LEN ST VarINT1 (*结果 Var1INT1 为 8*)
FBD	LEN: 'Hollysys'—STR; —VarINT1=8

4.1.2 LEFT——左边取字符串指令

- 功能：从字符串左边取字符串。
- 指令格式：LEFT（STR,SIZE），其中输入 STR 是 STRING 类型，为输入字符串，SIZE 是 INT 型，为从输入字符串左边开始获取的字符个数，输出数据类型是 STRING 型。

✧ 指令使用举例

变量声明：

VarSTRING: STRING;

POU 语言	程 序
LD	LEFT: EN; 'Hollysys'—STR; 3—SIZE; —VarSTRING='Hol'
ST	VarSTRING := LEFT (‘Hollysys’,3); (*结果为' Hol '*)
IL	LD ' Hollysys ' LEFT 3 ST VarSTRING (*结果为' Hol '*)
FBD	LEFT: 'Hollysys'—STR; 3—SIZE; —VarSTRING='Hol'

4.1.3 RIGHT——右边取字符串指令

- 功能：从字符串右边取字符串。
- 指令格式：RIGHT（STR,SIZE），其中输入 STR 是 STRING 类型，为输入字符串，SIZE 是 INT 型，为从字符串右边开始获取的字符个数，输出数据类型是 STRING 型。
  - ✧ 指令使用举例

    变量声明：

    VarSTRING: STRING;

POU 语言	程　序
LD	RIGHT: EN; 'Hollysys'–STR; 3–SIZE; —VarSTRING='sys'
ST	VarSTRING := RIGHT(‘Hollysys’,3);　(*结果为' sys '*)
IL	LD　' Hollysys ' RIGHT 3 ST　VarSTRING　(*结果为' sys '*)
FBD	RIGHT: 'Hollysys'–STR; 3–SIZE; —VarSTRING='sys'

4.1.4 MID——中间取字符串指令

- 功能：从字符串中间取字符串。
- 指令格式：MID(STR,LEN,POS)，其中输入 STR 是 STRING 类型，为输入字符串。LEN 和 POS 是 INT 型，该指令从 POS 开始从左往右获取 LEN 个字符，输出数据类型是 STRING 型。
  - ✧ 指令使用举例

    变量声明：

    VarSTRING: STRING;

POU 语言	程　序
LD	MID: EN; 'Hollysys'–STR; 2–LEN; 4–POS; —VarSTRING='ly'
ST	VarSTRING:= MID(‘Hollysys’,2,4);　(*结果为‘ly’ *)
IL	LD　‘Hollysys’ RIGHT　2,4 ST　VarSTRING　(*结果为‘ly’ *)
FBD	MID: 'Hollysys'–STR; 2–LEN; 4–POS; —VarSTRING='ly'

4.1.5 CONCAT——合并字符串指令

◆ 功能：把两个字符串按前后顺序结合成一个字符串，输入和输出都是 STRING 型。

✧ 指令使用举例

变量声明：

VarSTRING: STRING;

POU 语言	程　序
LD	CONCAT: EN; 'Holly'–STR1; 'sys'–STR2 ——VarSTRING='Hollysys'
ST	VarSTRING := CONCAT ('Holly', 'sys'); (*结果为'Hollysys'*)
IL	LD 'Holly' CONCAT ‘sys’ ST VarSTRING (*结果为'Hollysys'*)
FBD	CONCAT: 'Holly'–STR1; 'sys'–STR2 ——VarSTRING='Hollysys'

4.1.6 INSERT——插入字符串指令

◆ 功能：把一个字符串插入到另一个字符串中。

◆ 指令格式：INSERT(STR1,STR2,POS)。输入 STR1 和 STR2 是 STRING 类型，POS 是 INT 型，该指令把 STR2 插入到 STR1 的 POS 位置之后。输出数据类型是 STRING 型。

✧ 指令使用举例

变量声明：

VarSTRING: STRING;

POU 语言	程　序
LD	INSERT: EN; 'Hoysys'–STR1; 'll'–STR2; 2–POS ——VarSTRING='Hollysys'
ST	VarSTRING := INSERT ('Hoysys', ‘ll’,2); (*结果为 'Hollysys'*)
IL	LD 'Hoysys' INSERT ‘ll’,2 ST VarSTRING (*结果为 'Hollysys'*)
FBD	INSERT: 'Hoysys'–STR1; 'll'–STR2; 2–POS ——VarSTRING='Hollysys'

4.1.7 DELETE——删除字符指令

- 功能：从字符串中删除字符。
- 指令格式：DELETE(STR,LEN,POS)。输入 STR 是 STRING 类型，为输入字符串。LEN 和 POS 是 INT 型，该指令从输入字符的位置 POS 处开始从左往右删除 LEN 个字符，输出数据类型是 STRING 型。

  ✧ 指令使用举例

  变量声明：

  VarSTRING: STRING;

POU 语言	程　序
LD	DELETE: EN; 'Hollysys'–STR; 3–LEN; 6–POS; —VarSTRING='Holly'
ST	VarSTRING := DELETE ('Hollysys',3,6);　(*结果为'Holly'*)
IL	LD　'Hollysys' DELETE　3,6 ST　VarSTRING　(*结果为'Holly'*)
FBD	DELETE: 'Hollysys'–STR; 3–LEN; 6–POS; —VarSTRING='Holly'

4.1.8 REPLACE——替换字符串指令

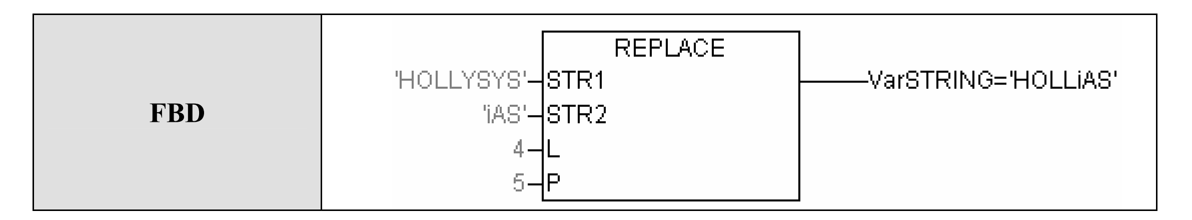
- 功能：用一个字符串替代另一字符串中的部分内容。
- 指令格式：REPLACE(STR1,STR2,L,P)。输入 STR1 和 STR2 是 STRING 类型，为输入字符串。L 和 P 是 INT 型，该指令用 STR2 代替 STR1 中从 P 位置开始的 L 个字符。输出数据类型是 STRING 型。

  ✧ 指令使用举例

  变量声明：

  VarSTRING: STRING;

POU 语言	程　序
LD	REPLACE: EN; 'HOLLYSYS'–STR1; 'iAS'–STR2; 4–L; 5–P; —VarSTRING='HOLLiAS'
ST	VarSTRING:= REPLACE ('HOLLYSYS', 'iAS',4,5); (*结果为'HOLLiAS'*)
IL	LD　'HOLLYSYS' REPLACE　'iAS', 4,5 ST　VarSTRING　(*结果为'HOLLiAS'*)

FBD	

4.1.9 FIND——查找字符串指令

- **功能**：在一个字符串中查找与另一字符串完全相同的内容。
- **指令格式**：FIND(STR1,STR2)。输入 STR1 和 STR2 都是 STRING 类型，为输入字符串，返回数据类型为 INT 型。指令功能为在第一个字符串 STR1 中查找与字符串 STR2 完全相同的部分，返回该相同部分在字符串 STR1 的起始位置。若没有完全相同的部分，输出结果为 0。
  - ✧ **指令使用举例**
    变量声明：
    VarINT1: INT;

POU 语言	程　序
LD	FIND: EN; 'HOLLYSYS'–STR1; 'SYS'–STR2; VarINT1=6
ST	VarINT1 := FIND ('HOLLYSYS', 'SYS'); (*结果为 6*)
IL	LD 'HOLLYSYS' FIND 'SYS' ST VarINT1 (*结果为 6*)
FBD	FIND: 'HOLLYSYS'–STR1; 'SYS'–STR2; VarINT1=6

4.2 BCD 码转换指令（Util.lib）

BCD 码的一个字节包含 0 到 99 之间的整数。每个十进制位对应 4 位，十位数存储在 4-7 位，个位数存储在 0-3 位。BCD 码格式和 16 进制表达方式很相似，差别在于 BCD 字节值是 0-99，而 16 进制是 0-FF。

BCD 码用 4 位二进制数表示一个十进制数位，整个十进制数用一串 BCD 码来表示。例如，十进制数 59 表示成 BCD 码为 0101 1001，但表示成 2 进制为 2#111011。51 转换成 BCD 码，5 的二进制是 0101，1 的二进制是 0001，BCD 码字节为 0101 0001。

4.2.1 BCD_TO_INT——BCD 码转整型指令

- 功能：该指令将 BCD 码转为 INT 值。
- 输入输出类型

  输入 B：BYTE 型，输入 BCD 码的二进制形式（或者该二进制对应的十进制和十六进制）。比如 BCD 码 49，表示为 2#100_1001（或者对应 10#73、16#49），则此处输入 2#100_1001（或者 10#73，16#49）。

  输出：INT 型，该 BCD 码所代表的实际值，如果输入的字节不是 BCD 码，输出是-1。

✧ 指令使用举例

变量声明：

Varint1: INT;
Varint2: INT;
Varint3: INT;

POU 语言	程　序
LD	BCD_TO_INT: EN, 73─B ──Varint1=49 BCD_TO_INT: EN, 151─B ──Varint2=97 BCD_TO_INT: EN, 15─B ──Varint3=-1
ST	Varint1:=BCD_TO_INT(73); (*结果为 49 *) Varint2:=BCD_TO_INT(151); (*结果为 97*) Varint3:=BCD_TO_INT(15); (*输出-1，因为不是 BCD 码格式*)
IL	LD 73 BCD_TO_INT ST Varint1 (*结果为 49 *)
	LD 151 BCD_TO_INT ST Varint2 (*结果为 97*)
	LD 15 BCD_TO_INT ST Varint3 (*输出-1，因为不是 BCD 码格式*)
FBD	BCD_TO_INT: 73─B ──Varint1=49 BCD_TO_INT: 151─B ──Varint2=97 BCD_TO_INT: 15─B ──Varint3=-1

4.2.2 INT_TO_BCD——整型转 BCD 码指令

◆ 功能：该指令将整数值转换成 BCD 码格式。当整数值不能转换成 BCD 码字节时，输出数值 255。

◆ 输入输出类型

输入 I：INT 型，如果整数值为 49，则此处输入整型数据 49。

输出：BYTE 型，转换完的 BCD 码的值，比如将 49 转换为 BCD 码为 2#100_1001，则此处输出 2#100_1001（或者该二进制对应的 10#73、16#49）。

✧ 指令使用举例

变量声明：

Varbyte1: BYTE;
Varbyte2: BYTE;
Varbyte3: BYTE;

<table>
<tr><th>POU 语言</th><th>程　序</th></tr>
<tr><td>LD</td><td>INT_TO_BCD: EN; I=49 → Varbyte1=73
INT_TO_BCD: EN; I=97 → Varbyte2=151
INT_TO_BCD: EN; I=100 → Varbyte3=255</td></tr>
<tr><td>ST</td><td>Varbyte1:=INT_TO_BCD(49); (*结果为 73*)
Varbyte2:= INT_TO_BCD(97); (*结果为 151*)
Varbyte3:= INT_TO_BCD(100); (*错误！输出：255*)</td></tr>
<tr><td rowspan="3">IL</td><td>LD 49
INT_TO_BCD
ST Varbyte1 (*结果为 73*)</td></tr>
<tr><td>LD 97
INT_TO_BCD
ST Varbyte2 (*结果为 151*)</td></tr>
<tr><td>LD 100
INT_TO_BCD
ST Varbyte3 (*错误！输出：255*)</td></tr>
<tr><td>FBD</td><td>INT_TO_BCD: I=49 → Varbyte1=73
INT_TO_BCD: I=97 → Varbyte2=151
INT_TO_BCD: I=100 → Varbyte3=255</td></tr>
</table>

4.3 位转换指令（Util.lib）

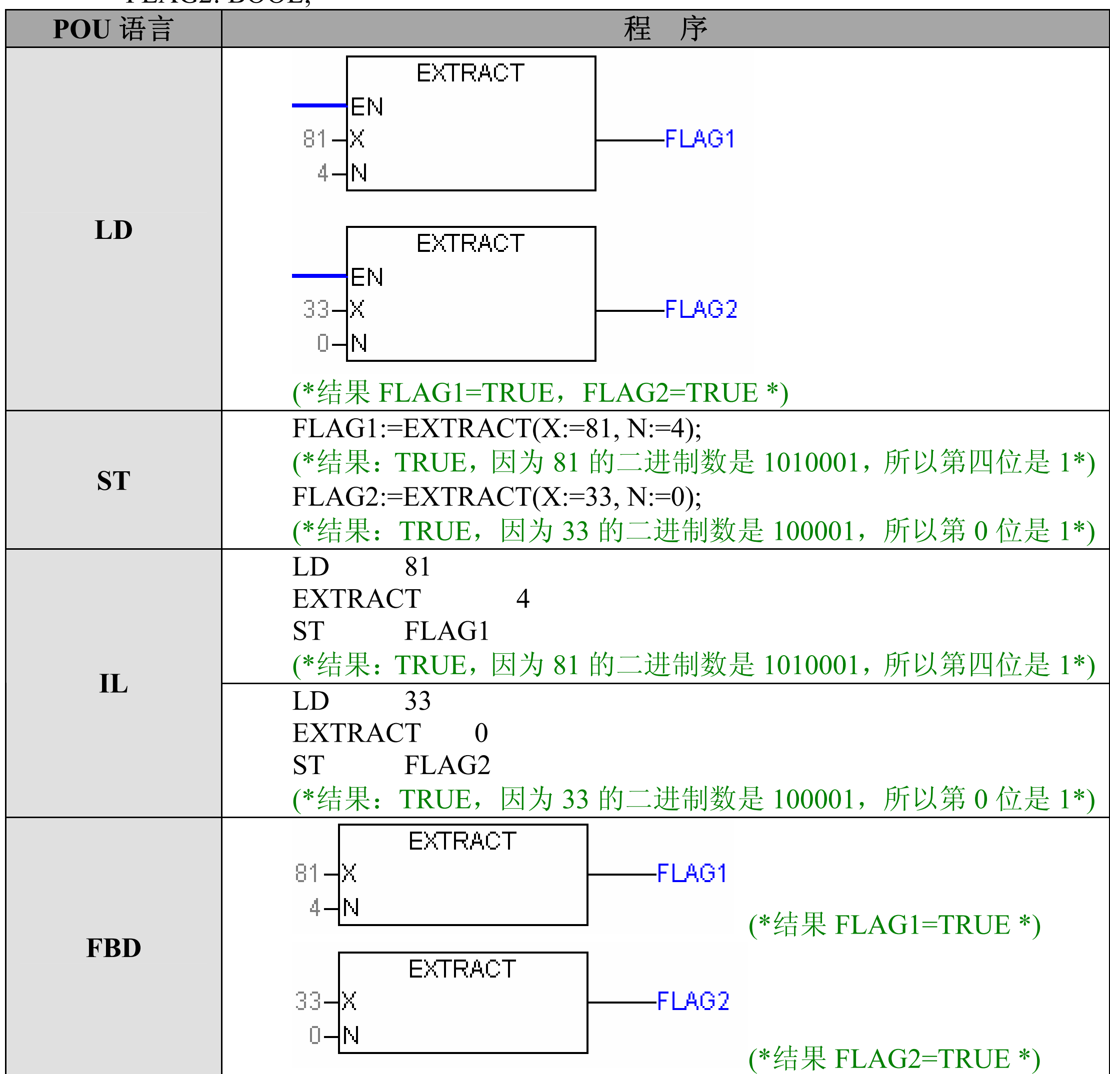
4.3.1 EXTRACT——位提取指令

◆ 功能：输入变量 X 是 DWORD 类型，N 是 BYTE 型。输出是 BOOL 类型的值，其值为输入 X 的第 N 位。指令从 X 的第 0 位开始算起。

✧ 指令使用举例

变量声明：

FLAG1: BOOL;
FLAG2: BOOL;



POU 语言	程　序
LD	EXTRACT: EN; 81–X; 4–N; 输出 –FLAG1 EXTRACT: EN; 33–X; 0–N; 输出 –FLAG2 (*结果 FLAG1=TRUE，FLAG2=TRUE *)
ST	FLAG1:=EXTRACT(X:=81, N:=4); (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*) FLAG2:=EXTRACT(X:=33, N:=0); (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
IL	LD 81 EXTRACT 4 ST FLAG1 (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*)
	LD 33 EXTRACT 0 ST FLAG2 (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
FBD	EXTRACT: 81–X; 4–N; 输出 –FLAG1　(*结果 FLAG1=TRUE *) EXTRACT: 33–X; 0–N; 输出 –FLAG2　(*结果 FLAG2=TRUE *)

4.3.2 PACK——位整合指令

◆ 功能：能把输入位 B0、B1、……、B7 合成为一个字节，和这个指令相对应的指令是 UNPACK。

✧ 指令使用举例

变量声明：

Varbyte1: BYTE;

POU 语言	程　序
LD	PACK EN 0–B0 1–B1 0–B2 1–B3 1–B4 0–B5 1–B6 0–B7 —Varbyte1=2#01011010
ST	Varbyte1:= PACK(0,1,0,1,1,0,1,0);　(*结果为 2#01011010*)
IL	LD　FALSE PACK　TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE ST　Varbyte1　(*结果为 2#01011010*)
FBD	PACK 0–B0 1–B1 0–B2 1–B3 1–B4 0–B5 1–B6 0–B7 —Varbyte1=2#01011010

4.3.3 PUTBIT——位赋值指令

◆ 功能：输入变量有 DWORD 型 X、BYTE 型 N 和 BOOL 型 B。PUTBIT 设置 X 值的第 N 位为值 B，从第 0 位算起。

✧ 指令使用举例

变量声明：

Var1: DWORD;
Var2: DWORD;

POU 语言	程　序
LD	PUTBIT EN 38–X 4–N TRUE–B —Var1=54
ST	Var2:=38;　(*二进制 100110*) Var1:=PUTBIT(Var2,4,TRUE);　(*结果：54 = 2#110110*)
IL	LD　38　(*二进制 100110*) PUTBIT　4,TRUE ST　Var1　(*结果：54 = 2#110110*)
FBD	PUTBIT 38–X 4–N TRUE–B —Var1=54

4.3.4 UNPACK——位拆分

◆ 功能：UNPACK 把字节型的输入 B 拆分转换成 8 个 BOOL 类型的输出变量 B0,…,B7，与 PACK 相反。

注意

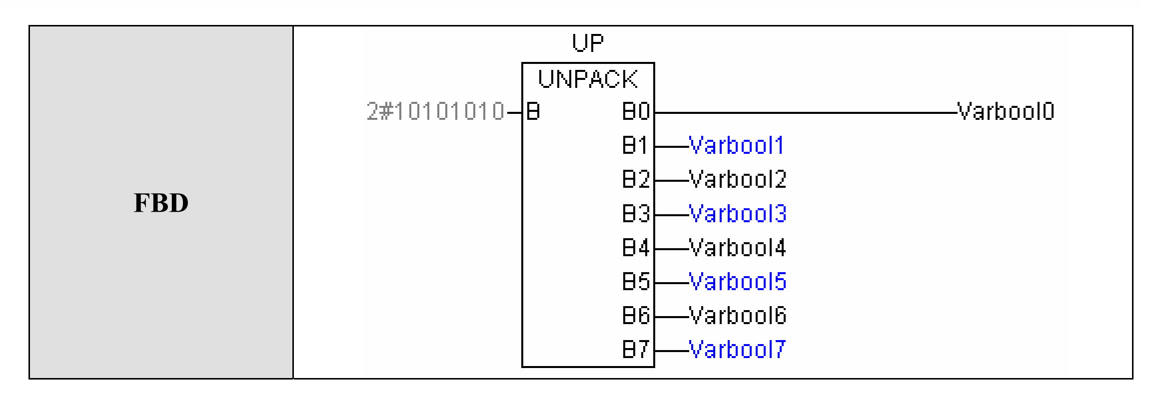
➢ 严格来讲，UNPACK 是一个内部功能块，因功能跟上述所述指令相近，故放在本节讲述。

✧ 指令使用举例

变量声明：

```
UP: UNPACK;
Varbool0: BOOL;
Varbool1: BOOL;
Varbool2: BOOL;
Varbool3: BOOL;
Varbool4: BOOL;
Varbool5: BOOL;
Varbool6: BOOL;
Varbool7: BOOL;
```

POU 语言	程　序
LD	UP UNPACK EN 2#10101010—B　B0—Varbool0 B1—Varbool1 B2—Varbool2 B3—Varbool3 B4—Varbool4 B5—Varbool5 B6—Varbool6 B7—Varbool7
IL	CAL　UP(B := 2#10101010) LD　UP.B1 ST　Varbool1 LD　UP.B2 ST　Varbool2 LD　UP.B3 ST　Varbool3 LD　UP.B4 ST　Varbool4 LD　UP.B5 ST　Varbool5 LD　UP.B6 ST　Varbool6 LD　UP.B7 ST　Varbool7 LD　UP.B0 ST　Varbool0

FBD	

UP
UNPACK
2#10101010
B
B0
B1
B2
B3
B4
B5
B6
B7
Varbool0
Varbool1
Varbool2
Varbool3
Varbool4
Varbool5
Varbool6
Varbool7

第5章 PowerPro 内部功能块

5.1 信号发生器功能块（Util.lib）

5.1.1 BLINK——脉冲信号发生器

◆ 功能：该功能块产生脉冲信号。ENABLE 和 OUT 是 BOOL 类型，TIMELOW 和 TIMEHIGH 是 TIME 类型。当 ENABLE 为 TRUE 时，脉冲信号发生器功能块 BLINK 开始工作，输出高电平时间为 TIMEHIGH，然后输出低电平时间为 TIMELOW，周期循环输出。

✧ 功能块使用举例

变量声明：

BLINKInst: BLINK;
Varbool1: BOOL;

POU 语言	程　序
LD	BLINKInst BLINK ENABLE　OUT T#5S–TIMELOW T#2s–TIMEHIGH
IL	CAL　BLINKInst(ENABLE := EN, TIMELOW := T#5s, TIMEHIGH := T#2s) LD　BLINKInst.OUT ST　Varbool1
FBD	BLINKInst BLINK ENABLE　OUT—Varbool1 T#5s–TIMELOW T#2s–TIMEHIGH

当功能块执行时，OUT 如图 5-1-1 所示波形输出。

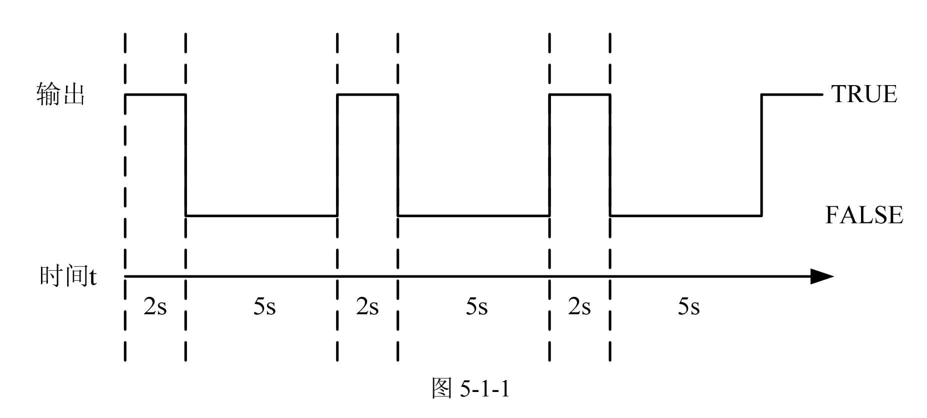


图 5-1-1

5.1.2 GEN——典型周期信号发生器

- **功能：**该功能块用于生成典型的周期信号，可以产生的周期信号有三角波、零起点三角波、上升锯齿波、下降锯齿波、方波、正弦波和余弦波共七种。
- **输入输出变量**
  - ✓ BASE：BOOL 型，定义循环周期。当 BASE 为 TRUE 时，信号发生器与定义的时间有关。当 BASE 为 FALSE 时，信号发生器与特定的循环次数有关。
  - ✓ PERIOD：TIME 型，定义了循环周期。
  - ✓ CYCLES：INT 型，发生的个数。
  - ✓ RESET：BOOL 型，当 RESET=TRUE 时，信号发生器被重新设置为 0。
  - ✓ AMPLITUDE：INT 型，定义了信号的振幅。
  - ✓ MODE 指定要产生的信号类型，MODE 的值可以是三角波 TRIANGLE、零起点三角波 TRIANGLE_POS、上升锯齿波 SAWTOOTH_RISE、下降锯齿波 SAWTOOTH_FALL、方波 RECTANGLE、正弦波 SINUS 和余弦波 COSINUS 等。直接在 MODE 处输入 TRIANGLE、TRIANGLE_POS、SAWTOOTH_RISE、SAWTOOTH_FALL、RECTANGLE、SINUS、COSINUS，则产生对应的波形。
  - ✓ OUT：INT 型，输出的变量。

✧ **功能块使用举例**

变量声明：

GENInst: GEN;
Var1: INT;

POU 语言	程　序
LD	GENInst GEN EN SINUS–MODE　　OUT——VAr1 TRUE–BASE t#3s–PERIOD 2–CYCLES 10–AMPLITUDE FALSE–RESET
FBD	GENInst GEN SINUS–MODE　　OUT——Var1 TRUE–BASE t#3s–PERIOD 2–CYCLES 10–AMPLITUDE FALSE–RESET

根据 MODE 处输入不同，产生的波形如图 5-1-2 所示。

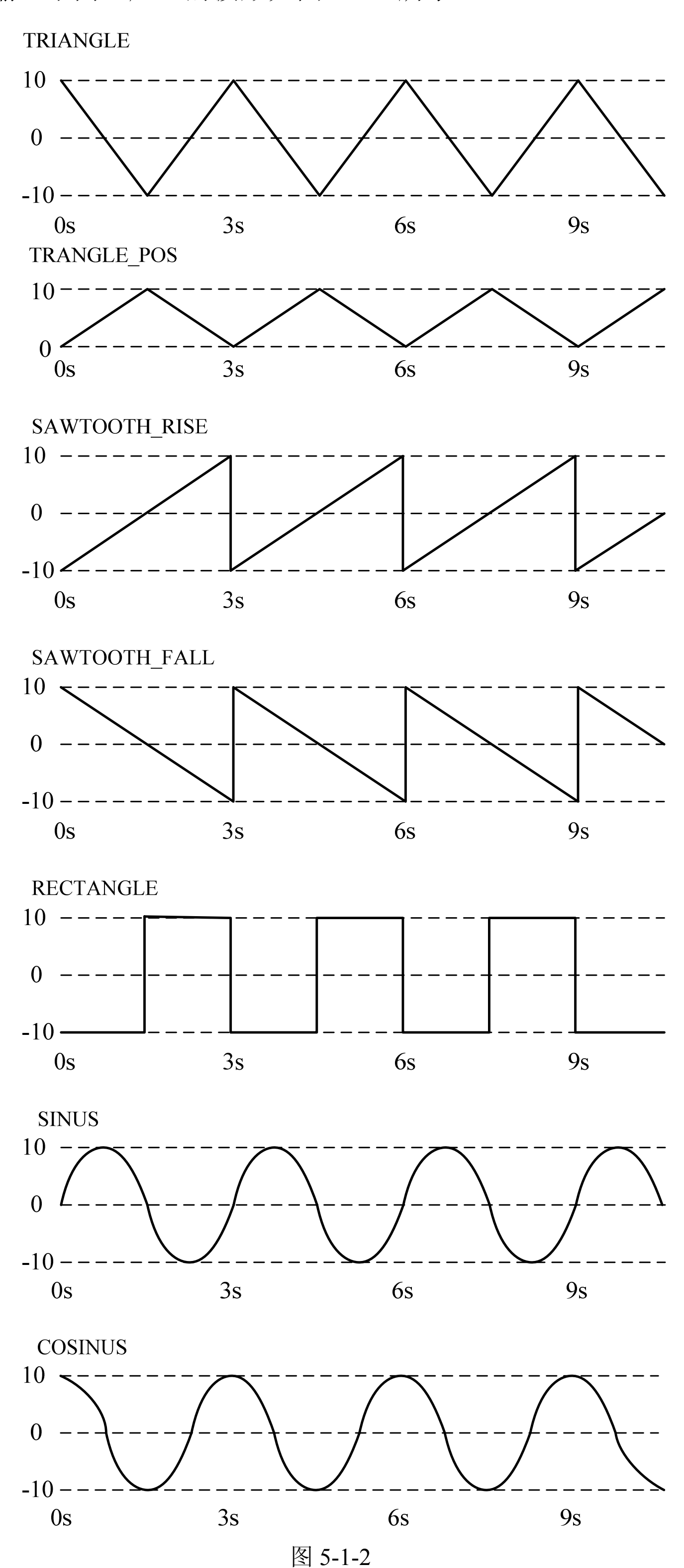


图 5-1-2

5.2 函数操纵器功能块（Util.lib）

5.2.1 CHARCURVE——特征曲线

- 功能：输入的 POINT 类型数组 P[0..N-1]在 XY 坐标图上定义了一条曲线，输入值 IN 为坐标图上的 X 轴上的点，输出值 OUT 为坐标图上该曲线所对应的 Y 轴的值。
- 输入输出变量
  - ✓ IN：INT 型，输入坐标图上 X 轴的值。N 是 BYTE 类型，指定义曲线所使用数组中的点数。P 为 ARRAY[0..n] OF POINT 型（2≤n≤11），用来在 XY 坐标上定义曲线。
  - ✓ POINT：基于两个整数值（X，Y）的结构。
  - ✓ ERR：BYTE 型，用于显示错误类型。

    ERR=1：数组中的点 P[0]..P[N-1]中的 X 值有错误。

    ERR=2：输入值 IN 不在 P[0].X 和 P[N-1].X 之间，即超出了数组定义曲线的 X 轴的范围。此时 OUT 输出 IN 包含在限制值 P[0].Y 和 P[N-1].Y 之间所对应的数据。

    ERR=4：输入 N 小于 2，或者大于 11。
  - ✓ OUT 是 INT 型，坐标图上曲线对应的 Y 轴的值。

✧ 功能块使用举例

变量声明：

```
CHARCURVEInst: CHARCURVE;
KL:ARRAY[0..10] OF POINT:=(X:=0,Y:=0),(X:=250,Y:=50),
                  (X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));
Var1: INT;
Varout: INT;
Varerr: BYTE;
```

POU 语言	程　序
LD	CHARCURVEInst CHARCURVE EN Var1 – IN　OUT – Varout 11 – N　ERR – Varerr KL – P　P –
IL	CAL　CHARCURVEInst(IN := Var1, N := 11, P := KL) LD　CHARCURVEInst.ERR ST　Varerr LD　CHARCURVEInst.OUT ST　Varout
FBD	CHARCURVEInst CHARCURVE Var1 – IN　OUT – Varout 11 – N　ERR – Varerr KL – P　P –

上例中，当功能块执行时，根据输入值的变化，对应的输出值如图 5-2-1，坐标曲线由数组 KL 确定，输入 IN 为 X 轴上的值，输出 OUT 为该曲线对应的 Y 轴上的值。

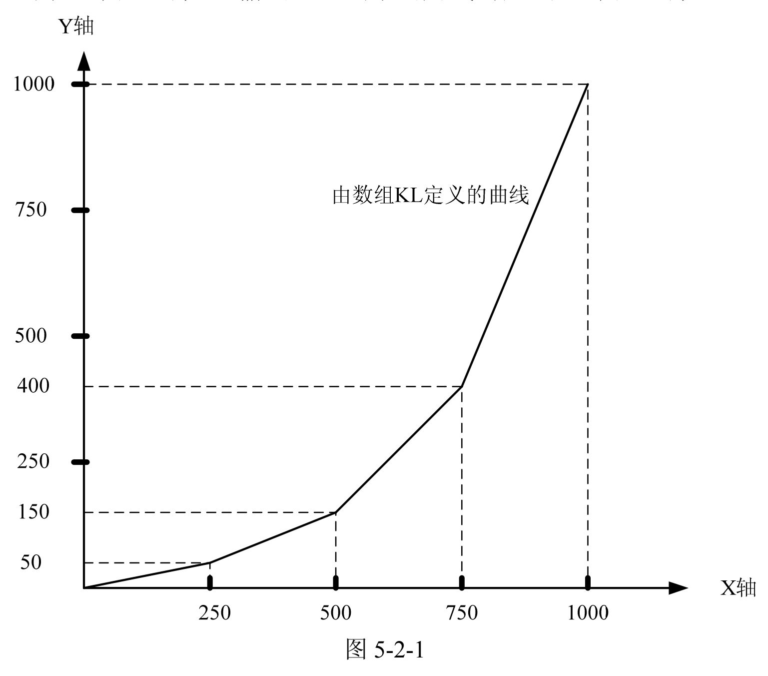


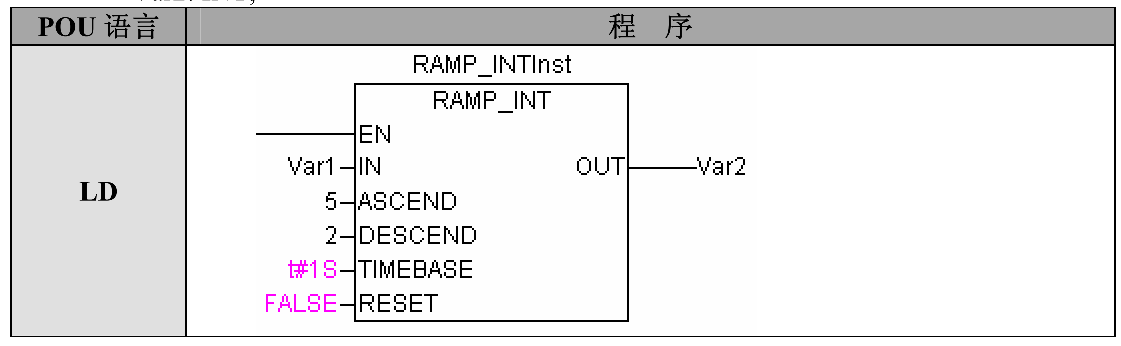
图 5-2-1

5.2.2 RAMP_INT——整型限速

- ◆ **功能：**RAMP_INT 限制输入函数的升降速度。
- ◆ **输入输出变量**
  - ✓ IN：INT 型，目标值。若当前设定值大于先前值，则按照设定的时间和上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的时间和下降速率进行减法运算，由 OUT 即时输出。
  - ✓ ASCEND：INT 型，在规定时间（TIMEBASE）内上升的数量。
  - ✓ DESCEND：INT 型，在规定时间（TIMEBASE）内下降的数量。
  - ✓ TIMEBASE：TIME 型，规定上升或者下降的速率。
  - ✓ RESET：BOOL 型，设置为 TRUE 时，RAMP_INT 被重新初始化。
  - ✓ OUT：INT 型，输出即时数据。
- ✧ **功能块使用举例**

  变量声明：

  RAMP_INTInst: RAMP_INT;
  Var1: INT;
  Var2: INT;

<table>
<tr><th>POU 语言</th><th>程　序</th></tr>
<tr><td>LD</td><td>
</td></tr>
</table>

IL	CAL RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE) LD RAMP_INTInst.OUT ST Var2
FBD	RAMP_INTInst RAMP_INT Var1 – IN OUT – Var2 5 – ASCEND 2 – DESCEND T#1s – TIMEBASE FALSE – RESET

上例中，当功能块执行时，根据输入值 IN 的变化，对应的输出值如图 5-2-2 所示。

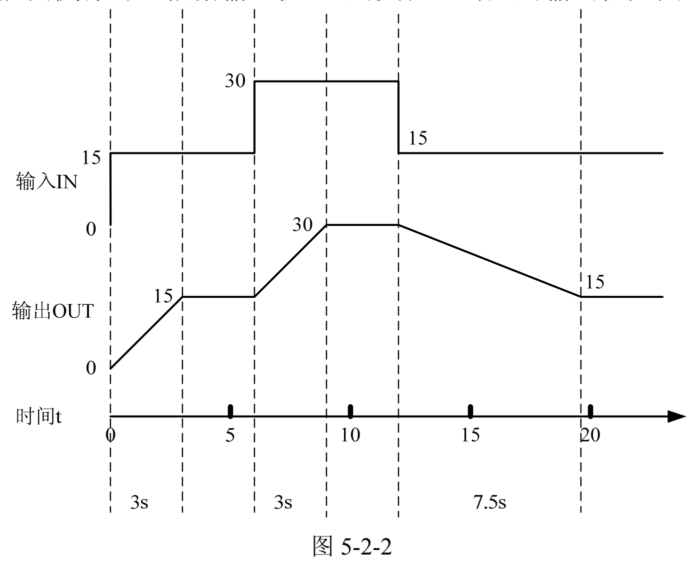


图 5-2-2

5.2.3 RAMP_REAL——实型限速

◆ **功能：**与 RAMP_INT 功能块一样。区别在于 RAMP_REAL 功能块的输入和输出都是 REAL 型数据。

请参见 RAMP_INT 功能块所述。

5.3 模拟量处理功能块（Util.lib）

5.3.1 HYSTERESIS——滞后

◆ 功能：该功能块的输入包括三个 INT 类型的数值 IN、HIGH 和 LOW。一个输出 OUT 是 BOOL 型。如果 IN 小于下限值 LOW，OUT 为 TRUE，保持至 IN 大于上限值 HIGH，OUT 为 FALSE，保持至 IN 小于下限值 LOW，OUT 为 TRUE，保持至 IN 大于上限值 HIGH，OUT 为 FALSE，如此循环。

✧ 功能块使用举例

变量声明：

```
HYSTERESISInst: HYSTERESIS;
Varool1: BOOL;
VarIN: INT;
```

POU 语言	程　序
LD	HYSTERESISInst HYSTERESIS EN VarIN–IN　OUT——Varool1 60–HIGH 30–LOW
IL	CAL　HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30) LD　HYSTERESISInst.OUT ST　Varool1
FBD	HYSTERESISInst HYSTERESIS VarIN–IN　OUT——Varool1 60–HIGH 30–LOW

上例中，当功能块执行时，根据输入值的变化，对应的输出值如图 5-3-1 所示。

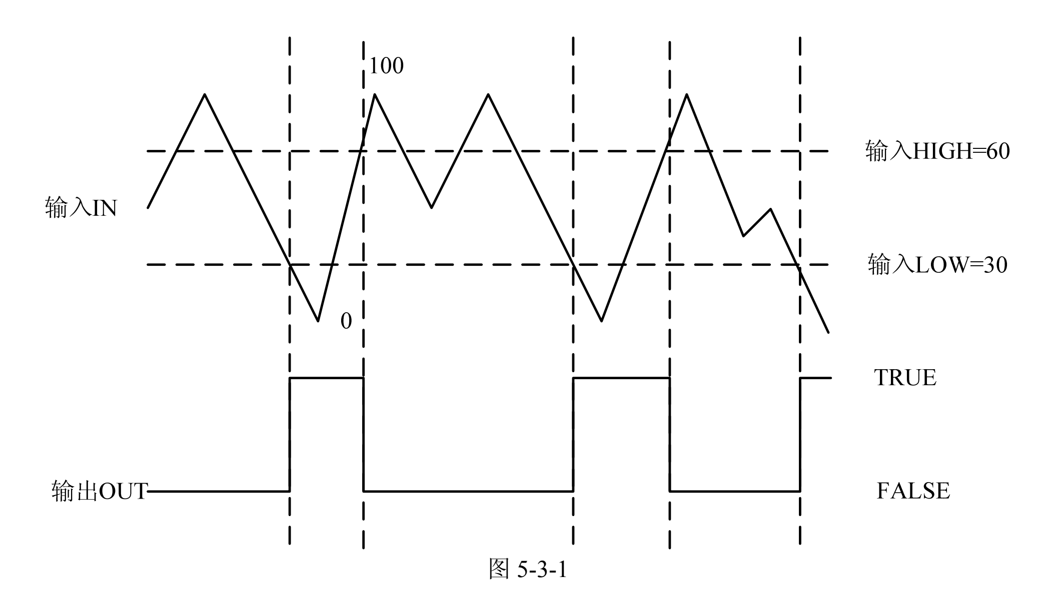


图 5-3-1

5.3.2 LIMITALARM——上下限报警

◆ 功能：输入 IN、HIGH 和 LOW 都是 INT 类型的数值，输出 O、U 和 IL 都是 BOOL 类型的数值。如果 IN 超出上限 HIGH，则 O 为 TRUE，U 和 IL 为 FALSE。如果 IN 低于下限 LOW，则 U 为 TRUE，O 和 IL 为 FALSE。如果 IN 在下限 LOW 和上限 HIGH 之间，则 IL 为 TRUE，O 和 U 为 FALSE。

✧ 功能块使用举例

变量声明：

```
LIMITALARMInst: LIMITALARM;
Var1: INT;
Varbool1: BOOL;
Varbool2: BOOL;
Varbool3: BOOL;
```

POU 语言	程　序
LD	LIMITALARMInst LIMITALARM EN Var1 – IN　O — Varbool1 60 – HIGH　U — Varbool2 30 – LOW　IL — Varbool3
IL	CAL　LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30) LD　LIMITALARMInst.U ST　Varbool2 LD　LIMITALARMInst.IL ST　Varbool3 LD　LIMITALARMInst.O ST　Varbool1
FBD	LIMITALARMInst LIMITALARM Var1 – IN　O — Varbool1 60 – HIGH　U — Varbool2 30 – LOW　IL — Varbool3

上例中，当功能块执行时，根据输入值的变化，对应的输出值如下图 5-3-2 所示。

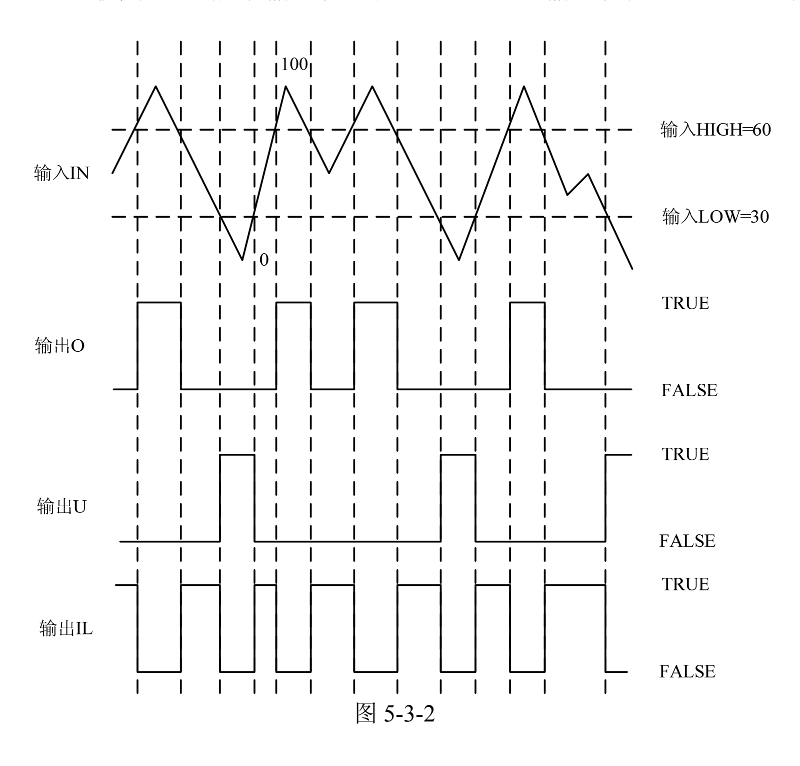


图 5-3-2

5.4 高等数学运算功能块（Util.lib）

5.4.1 DERIVATIVE——微分

◆ 功能：该功能块对连续输入的变量进行微分运算。为了获得最好结果，DERIVATIVE 功能块只对最新的四个输入值进行微分，使输入参数不精确产生的误差尽可能小。

◆ 输入输出类型
- ✓ IN：INT 或 REAL 型，连续输入的变量。
- ✓ TM：DWORD 型，微分时间（毫秒）。
- ✓ RESET：BOOL 型，复位信号。
- ✓ OUT：REAL 型，微分结果输出。

◆ 微分迭代公式

$$OUT = \frac{3*[IN(k)-IN(k-3)]+IN(k-1)-IN(k-2)}{3*TM(k-2)+4*TM(k-1)+3*TM(k)}$$

- ✓ k-3、k-2、k-1、k 为连续四次输入值的标记。

✧ 功能块使用举例

变量声明：

DERIVATIVEInst: DERIVATIVE;
Varreal1: REAL;
Varint1: INT;
VarBOOL1: BOOL;

POU 语言	程　序
LD	DERIVATIVEInst DERIVATIVE EN Varint1 – IN　OUT —— Varreal1 100 – TM VarBOOL1 – RESET
IL	CAL　DERIVATIVEInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD　DERIVATIVEInst.OUT ST　Varreal1
FBD	DERIVATIVEInst DERIVATIVE Varint1 – IN　OUT —— Varreal1 100 – TM VarBOOL1 – RESET

输入输出对应关系如图 5-4-1 所示。

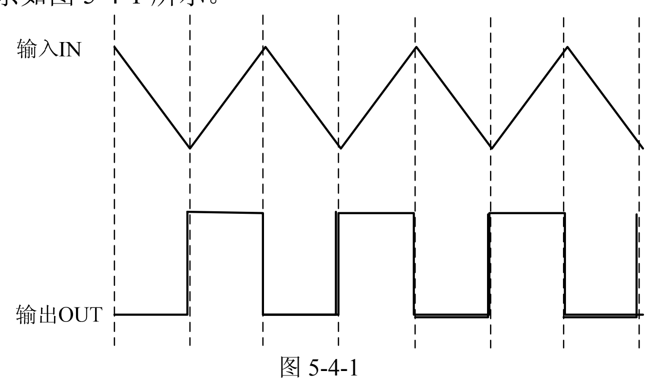


图 5-4-1

5.4.2 INTEGRAL——积分

- ◆ 功能：该功能块对连续输入的变量进行进行积分运算。
- ◆ 输入输出类型
  - ✓ IN：INT 或 REAL 型，连续输入的变量。
  - ✓ TM：DWORD 型，积分时间。
  - ✓ RESET：BOOL 型，复位信号，其值是 TRUE 时，重新启动功能块。
  - ✓ OUT：REAL 型，积分结果输出。
- ◆ 积分迭代公式

$$A(k) = A(k-1) + TM * IN(k-1)$$

$$B(k) = B(k-1) + TM * IN(k)$$

$$OUT(k) = \frac{A(k) + B(k)}{2}$$

  - ✓ k-1 、k 为连续两次输入值的标记。

✧ 功能块使用举例

变量声明：

```
INTEGRALInst: INTEGRAL;
Varreal1: REAL;
```

Varint1: INT;
VarBOOL1: BOOL;

POU 语言	程　序
LD	INTEGRALInst INTEGRAL EN Varint1 – IN　OUT — Varreal1 100 – TM VarBOOL1 – RESET
IL	CAL　INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD　INTEGRALInst.OUT ST　Varreal1
FBD	INTEGRALInst INTEGRAL Varint1 – IN　OUT — Varreal1 100 – TM VarBOOL1 – RESET

输入输出对应关系如下图 5-4-2 所示。

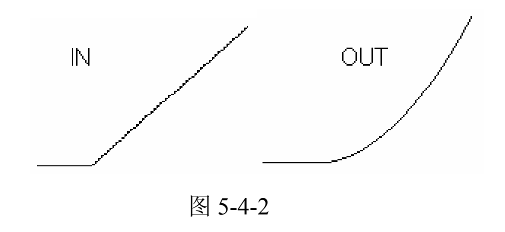


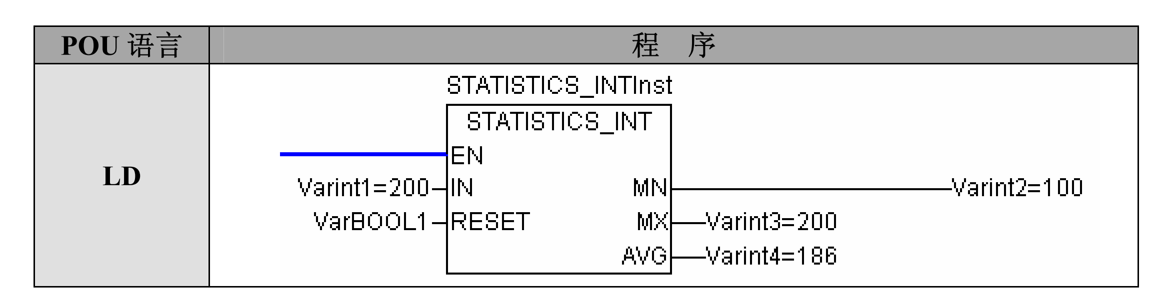
图 5-4-2

5.4.3 STATISTIC_INT——整型统计

- **功能**：对输入进行标准统计值的计算。输入 IN 的类型是 INT，当逻辑输入 RESET 是 TRUE 时，所有值重新初始化。MN，MX，AVG 分别代表 IN 的最小值、最大值和平均值。所有输出都是 INT 型。
  - ✧ **功能块使用举例**

    变量声明：

    STATISTICS_INTInst: STATISTICS_INT;
    VarBOOL1: BOOL;
    Varint1: INT;
    Varint2: INT;
    Varint3: INT;
    Varint4: INT;



POU 语言	程　序
LD	STATISTICS_INTInst STATISTICS_INT EN Varint1=200 – IN　MN — Varint2=100 VarBOOL1 – RESET　MX — Varint3=200 AVG — Varint4=186

IL	CAL STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1) LD STATISTICS_INTInst.MX ST Varint3 LD STATISTICS_INTInst.AVG ST Varint4 LD STATISTICS_INTInst.MN ST Varint2
FBD	STATISTICS_INTInst STATISTICS_INT Varint1=200—IN MN—Varint2=100 VarBOOL1—RESET MX—Varint3=200 AVG—Varint4=186

5.4.4 STATISTIC_REAL——实型统计

◆ 功能：计算一些标准统计值。输入 IN 的类型是 REAL，当逻辑输入 RESET 是 TRUE 时，所有值重新初始化。MN，MX，AVG 分别代表 IN 的最小值、最大值和平均值。所有输出都是 REAL 型。

✧ 功能块使用举例

变量声明：

STATISTICS_REALInst: STATISTICS_REAL;
VarBOOL1: BOOL;
Varreal1: REAL;
Varreal2: REAL;
Varreal3: REAL;
Varreal4: REAL;

POU 语言	程　序
LD	STATISTICS_REALInst STATISTICS_REAL —EN Varreal1=200—IN MN—Varreal2=100 VarBOOL1—RESET MX—Varreal3=200 AVG—Varreal4=187.1985
IL	CAL STATISTICS_REALInst(IN := Varreal1, RESET := VarBOOL1) LD STATISTICS_REALInst.MX ST Varreal3 LD STATISTICS_REALInst.AVG ST Varreal4 LD STATISTICS_REALInst.MN ST Varreal2
FBD	STATISTICS_REALInst STATISTICS_REAL Varreal1=200—IN MN—Varreal2=100 VarBOOL1—RESET MX—Varreal3=200 AVG—Varreal4=189.2663

注意

➢ 该功能块与 STATISTIC_INT 功能相同，只是输入和输出为 REAL 型。

5.4.5 VARIANCE——平方偏差

- **功能**：该功能块计算变量输入值的平方偏差。输入 IN 是 REAL 类型，RESET 是 BOOL 类型，输出 OUT 是 REAL 类型。当 RESET=TRUE 时，该功能块复位。
  - ✓ 标准偏差可以由平方偏差的平方根得到。
  - ✧ 功能块使用举例

    变量声明：

        VARIANCEInst: VARIANCE;
        VarBOOL1: BOOL;
        Varreal1: REAL;
        Varreal2: REAL;

POU 语言	程　序
LD	VARIANCEInst VARIANCE EN Varreal1 – IN　OUT – Varreal2 VarBOOL1 – RESET
IL	CAL　VARIANCEInst(IN := Varreal1, RESET := VarBOOL1) LD　VARIANCEInst.OUT ST　Varreal2
FBD	VARIANCEInst VARIANCE Varreal1 – IN　OUT – Varreal2 VarBOOL1 – RESET

5.5 PID 控制器功能块（Util.lib）

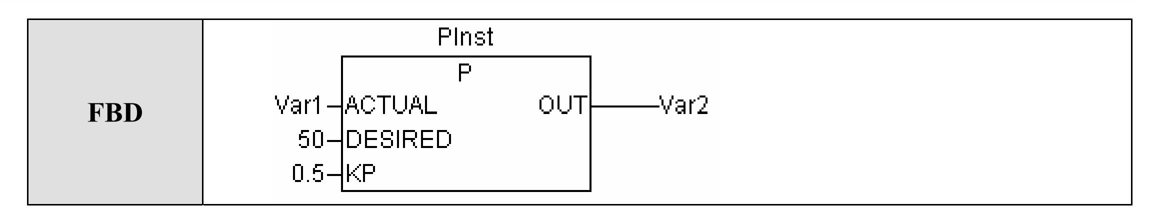
5.5.1 P——比例控制器

- **功能**：该功能块为比例控制器。输入的测量值 ACTUAL、设定值 DESIRED 和比例因子 KP 都是 REAL 型。输出值 OUT 是 REAL 型。
- **控制方程**：OUT=ACTUAL+（DESIRED-ACTUAL）*KP
  - ✧ 功能块使用举例

    变量声明：

        PInst: P;
        Var1: REAL;
        Var2: REAL;

POU 语言	程　序
LD	PInst P EN Var1 – ACTUAL　OUT – Var2 50 – DESIRED 0.5 – KP
IL	CAL　PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5) LD　PInst.OUT ST　Var2



5.5.2 PD——比例微分控制器

- 功能：该功能块为比例微分控制器。
- 输入输出说明
  - ✓ ACTUAL：REAL 型，测量值。
  - ✓ SET_POINT：REAL 型，设定值。
  - ✓ KP：REAL 型，比例系数。
  - ✓ TV：DWORD 型，微分时间。
  - ✓ Y_MANUAL：REAL 型，手动值，MANUAL=TRUE 时，Y= Y_MANUAL。
  - ✓ Y_OFFSET：REAL 型，输出值的偏移量。
  - ✓ Y_MIN：REAL 型，输出值的最小值。
  - ✓ Y_MAX：REAL 型，输出值的最大值。
  - ✓ MANUAL：BOOL 型，TRUE 时为手动调节，FALSE 时为自动调节。
  - ✓ RESET：BOOL 型，TRUE 时重置该控制器，正常运行时应置 FALSE。
  - ✓ Y：REAL 型，输出值。
  - ✓ LIMITS_ACTIVE：BOOL 型，输出值超限时等于 TRUE，即超出（Y_MIN，Y_MAX）。
- 控制方程

$$\Delta = \mathrm{SET}_POINT - ACTUAL$$

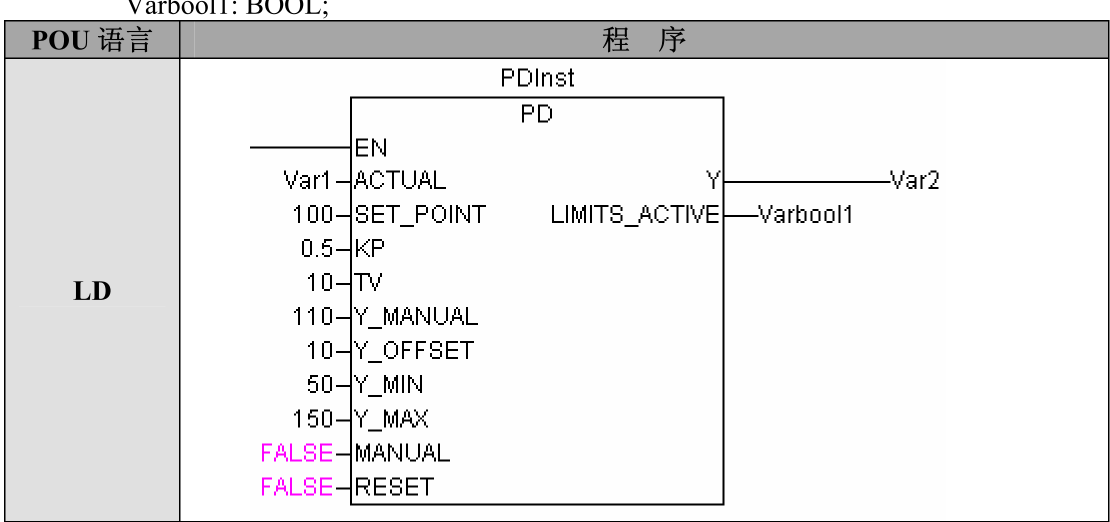
$$Y = KP * (\Delta + TV * \frac{\sigma\Delta}{\sigma t}) + Y_OFFSET$$

该功能块会自动计算 $\frac{\sigma\Delta}{\sigma t}$，用户无需考虑。

✧ 功能块使用举例

变量声明：

```
PDInst: PD;
Var1: REAL;
Var2: REAL;
Varbool1: BOOL;
```



IL	CAL PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE) LD PDInst.LIMITS_ACTIVE ST Varbool1 LD PDInst.Y ST Var2
FBD	PDInst PD Var1 — ACTUAL Y — Var2 100 — SET_POINT LIMITS_ACTIVE — Varbool1 0.5 — KP 10 — TV 110 — Y_MANUAL 10 — Y_OFFSET 50 — Y_MIN 150 — Y_MAX FALSE — MANUAL FALSE — RESET

调整波形如图 5-5-1 所示。

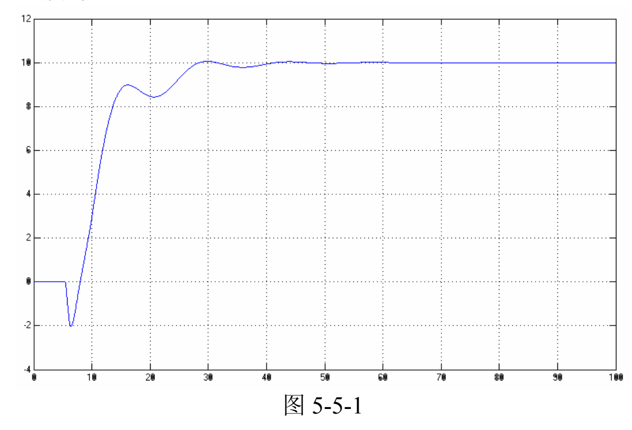


图 5-5-1

5.5.3 PID——比例积分微分控制器

◆ 功能：该功能块为比例积分微分控制器。
  ✓ 当 TV=0 时，PID 控制器作为 PI 控制器。

◆ 输入输出说明
- ✓ ACTUAL：REAL 型，测量值。
- ✓ SET_POINT：REAL 型，设定值。
- ✓ KP：REAL 型，比例系数。
- ✓ TN：DWORD 型，积分时间。
- ✓ TV：DWORD 型，微分时间。
- ✓ Y_MANUAL：REAL 型，手动值，MANUAL=TRUE 时，Y= Y_MANUAL。
- ✓ Y_OFFSET：REAL 型，输出值的偏移量。
- ✓ Y_MIN：REAL 型，输出值的最小值。
- ✓ Y_MAX：REAL 型，输出值的最大值。
- ✓ MANUAL：BOOL 型，TRUE 时为手动调节，FALSE 时为自动调节。
- ✓ RESET：BOOL 型，TRUE 时重置该控制器，正常运行时应置 FALSE。

- ✓ Y：REAL 型，输出值。
- ✓ LIMITS_ACTIVE：BOOL 型，输出值超限时等于 TRUE，即超出（Y_MIN，Y_MAX）。
- ✓ OVERFLOW：BOOL 型，积分溢出时等于 TRUE。

◆ 控制方程

$$Y = KP * (\Delta + \frac{1}{TN} * \int \Delta(t)dt + TV * \frac{\sigma\Delta}{\sigma t}) + Y_OFFSET$$

$$\Delta = SET_POINT - ACTUAL$$

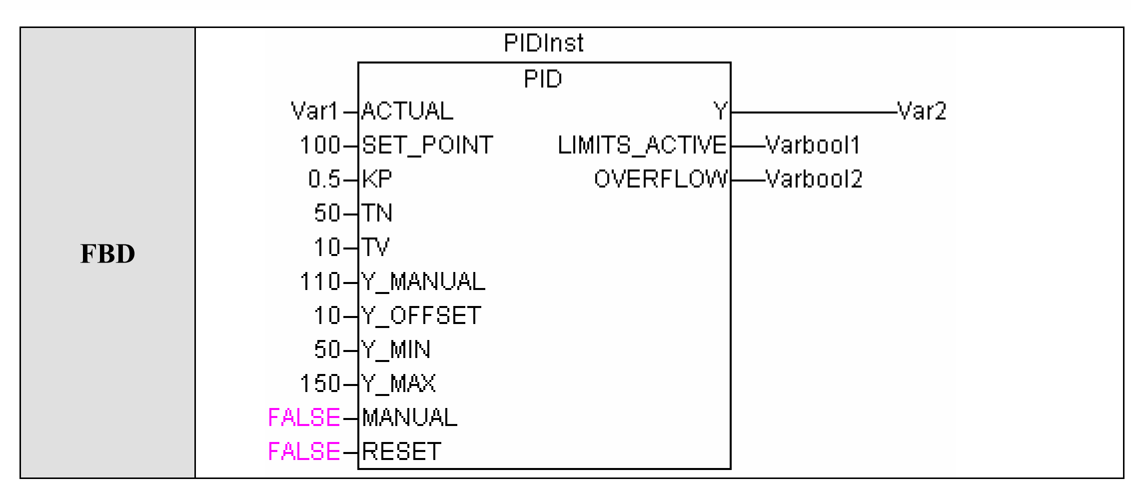
该功能块会自动计算 $\int \Delta(t)dt$ 与 $\frac{\sigma\Delta}{\sigma t}$，用户无需考虑。

✧ 功能块使用举例

变量声明：

```
PIDInst: PID;
Var1: REAL;
Var2: REAL;
Varbool1: BOOL;
Varbool2: BOOL;
```

POU 语言	程 序
LD	PIDInst PID —EN Var1—ACTUAL　　Y—Var2 100—SET_POINT　　LIMITS_ACTIVE—Varbool1 0.5—KP　　OVERFLOW—Varbool2 50—TN 10—TV 110—Y_MANUAL 10—Y_OFFSET 50—Y_MIN 150—Y_MAX FALSE—MANUAL FALSE—RESET
IL	CAL PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE) LD PIDInst.LIMITS_ACTIVE ST Varbool1 LD PIDInst.OVERFLOW ST Varbool2 LD PIDInst.Y ST Var2



调整波形如图 5-5-2 所示。

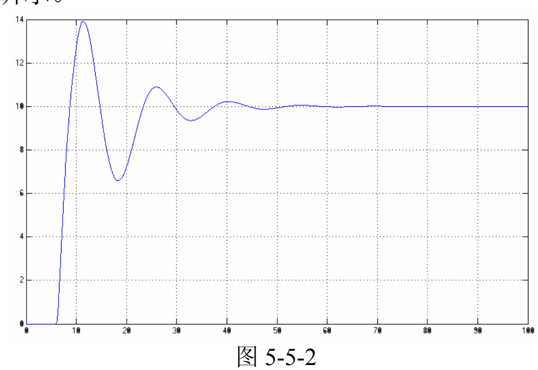


图 5-5-2

5.6 PID2 运算功能块（Hollysys_PLC_Util.lib）

5.6.1 PID2——PID 运算

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	PID2 RESET MVn L ALARM ACT PVn SV KP TS TI TD KD I_INCREASE I_DECREASE O_INCREASE O_DECREASE O_UpperLimit O_LowerLimit

◆ 输入输出说明

输入	功能	数据类型	值
RESET	复位，高电平有效	BOOL	
L	滤波系数（0-1）	REAL	
ACT	动作方向/报警设置 （ACT 的.2 和.5 不能同时为 1）	UINT	.0 = 0—正动作，1—逆动作 .1 = 0—输入报警无效， 1—输入报警有效 .2 = 0—输出报警无效， 1—输出报警有效 .5 = 0—输出上下限设定无效， 1—输出上下限设定有效
PVn	测定值	REAL	
SV	设定值	REAL	
KP	比例增益（KP>0）	REAL	
TS	采样周期（TS>0）	REAL	
TI	积分常数（TI≥0）	REAL	
TD	微分常数（TD≥0）	REAL	
KD	微分增益（0-1）	REAL	
I_INCREASE	输入变化量增侧报警设定值	REAL	
I_DECREASE	输入变化量减侧报警设定值	REAL	
O_INCREASE	输出变化量增侧报警设定值	REAL	
O_DECREASE	输出变化量减侧报警设定值	REAL	
O_UpperLimit	输出上限设定值	REAL	
O_LowerLimit	输出下限设定值	REAL	
输出	**功能**	**数据类型**	**值**
MVn	输出值	REAL	
ALARM	报警输出	UINT	.0 = 0—输入变化量增侧正常， 1—输入变化量增侧溢出 .1 = 0—输入变化量减侧正常， 1—输入变化量减侧溢出 .2 = 0—输出变化量增侧正常， 1—输出变化量增侧溢出 .3 = 0—输出变化量减侧正常， 1—输出变化量减侧溢出

◆ **PID** 基本运算式

表达式	含义	表达式	含义
EV_n	本次采样时的偏差	D_n	本次的微分项
EV_{n-1}	前一次采样时的偏差	D_{n-1}	前一次的微分项
SV	设定值	K_P	比例增益
PV_{nf}	本次采样时的测定值（滤波后）	T_S	采样周期
PV_{nf-1}	前一次采样时的测定值（滤波后）	T_I	积分时间常数
PV_{nf-2}	前二次采样时的测定值（滤波后）	T_D	微分时间常数
△MV	输出变化量	MV_n	本次的操作量
KD：	微分增益		

正动作：
$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$ $EV_n = PV_{nf} - SV$ $D_n = \frac{T_D}{T_S + KD * T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$
逆动作：
$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$ $EV_n = SV - PV_{nf}$ $D_n = \frac{T_D}{T_S + KD * T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$

PV_{nf} 是根据读入的测定值由下列运算式求得的值。

滤波后测定值 $PV_{nf} = PV_n + L(PV_{nf-1} - PV_n)$

PV_n：本次采样时的测定值。

L：滤波系数（0~1）。

PV_{nf-1}：1 个周期前的测定值（滤波后）。

5.7 Modbus 校验功能块（Hollysys_PLC_Modbus_CRC.lib）

5.7.1 Generate_CRC——Modbus 校验产生

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	GENERATE_CRC pData　CRC_Code byteCounter　FINISH

◆ 输入输出说明

输入	功能	数据类型	值
pData	需要校验的数据的起始指针	POINTER TO BYTE	
byteCounter	需要校验的数据的字节数	WORD	

输出	功能	数据类型	值
CRC_Code	校验结果	WORD	
FINISH	是否完成校验	BOOL	0：未完成校验 1: 完成校验

✧ **Generate_CRC 功能块举例（变量定义与梯形图）**

```
0002 VAR
0003     EN: BOOL;
0004     Example: Generate_CRC;
0005     pData: POINTER TO BYTE;
0006     DataCRC AT %MB60 : ARRAY [1..6] OF BYTE:=16#01,16#03,16#0C,16#1C,16#00,16#01;
0007     CRC_Result AT %MW66: WORD;
0008 END_VAR
```

0001: EN —| |— ADR (EN, DataCRC) — pData

0002: EN —| |— Example Generate_CRC (EN, pData — pData, 6 — byteCounter; CRC_Code — CRC_Result, FINISH)

梯形图描述:

EN 置位时，先用 ADR 算出 DataCRC 数组的指针，然后赋值给 Generate_CRC 功能块，在 byteCounter 处填入需要校验的字节数 6，则 CRC_Code 为得出的 16#01、16#03、16#0C、16#1C、16#00、16#01 的校验值，存储在%MW66 处。

注意

➢ 必须使用使能运算符调用该功能块。

第6章 PowerPro 外部 G3 功能块

6.1 双稳态功能块（Standard.lib）

6.1.1 SR——置位双稳态功能块

- 功能：置位双稳态功能块，置位优先。
- 逻辑关系：Q1=(NOT RESET AND Q1) OR SET1

  其中 SET1 为置位信号，RESET 为复位信号。输入变量 SET1 和 RESET 以及输出变量 Q1 都是 BOOL 数据类型。

  ✧ 功能块使用举例

  变量声明：

  SRInst : SR;
  VarBOOL1: BOOL;
  VarBOOL2: BOOL;
  VarBOOL 3: BOOL;

POU 语言	程　序
LD	SRInst SR EN VarBOOL1 SET1 Q1 VarBOOL3 VarBOOL2 RESET 说明：VarBOOL3＝（NOT VarBOOL2 AND VarBOOL3）OR VarBOOL1
ST	SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2); VarBOOL3 := SRInst.Q1 ;
IL	CALSRInst(SET1:= VarBOOL1, RESET:= VarBOOL2) LD SRInst.Q1 ST VarBOOL3
FBD	SRInst SR VarBOOL1 SET1 Q1 VarBOOL3 VarBOOL2 RESET

6.1.2 RS——复位双稳态功能块

- 功能：复位双稳态功能块，复位优先。
- 逻辑关系：Q1=NOT RESET1 AND (Q1 OR SET)

  其中 SET 为置位信号，RESET1 为复位信号。输入变量 SET 和 RESET1 以及输出变量 Q1 都是 BOOL 数据类型。

✧ 功能块使用举例

变量声明：

RSInst : RS;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL 3: BOOL;

POU 语言	程　序
LD	RSInst RS EN VarBOOL1 SET Q1 VarBOOL3 VarBOOL2 RESET1 说明：VarBOOL3＝NOT VarBOOL2 AND (VarBOOL3 OR VarBOOL1)
ST	RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2); VarBOOL3 := RSInst.Q1 ;
IL	CAL RSInst(SET := VarBOOL1, RESET1 := VarBOOL2) LD RSInst.Q1 ST VarBOOL3
FBD	RSInst RS VarBOOL1 SET Q1 VarBOOL3 VarBOOL2 RESET1

6.1.3 SEMA——软件信号灯

◆ 功能：软件信号灯的表达式为：BUSY=SEMA(CLAIM, RELEASE)

◆ 逻辑关系：

```
BUSY := X;
IF CLAIM THEN X:=TRUE;
ELSE IF RELEASE THEN BUSY := FALSE; X:= FALSE;
END_IF
```

X 是一个内部的 BOOL 变量，初始化为 FALSE。输入变量 CLAIM 和 RELEASE 以及输出变量 BUSY 是 BOOL 数据类型。

当软件信号灯 SEMA 被调用时，如果 BUSY 为 TRUE，这就表示已经给软件信号灯 SEMA 分配了值（SEMA 被调用时，CLAIM = TRUE）。如果 BUSY 为 FALSE，则 SEMA 没有被调用，或者已经被释放（调用时 RELEASE = TRUE）。

✧ 功能块使用举例

变量声明：

SEMAInst : SEMA;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL 3: BOOL;

POU 语言	程　序
LD	SEMAInst SEMA EN VarBOOL1 CLAIM BUSY VarBOOL3 VarBOOL2 RELEASE

ST	SEMAInst(CLAIM:= VarBOOL1 , RELEASE:=VarBOOL2); VarBOOL3 := SEMAInst.BUSY;
IL	CAL SEMAInst(CLAIM:=VarBOOL1, RELEASE:=VarBOOL2) LD SEMAInst.BUSY ST VarBOOL3
FBD	SEMAInst SEMA VarBOOL1 – CLAIM BUSY — VarBOOL3 VarBOOL2 – RELEASE

6.2 触发器功能块（Standard.lib）

触发器包含上升沿检测触发器 R_TRIG 和下降沿检测触发器 F_TRIG，分别用于检测上升沿和下降沿。

6.2.1 R_TRIG——上升沿检测触发器

◆ 功能：用于检测上升沿，功能块源程序定义如下：

```
FUNCTION_BLOCK R_TRIG
VAR_INPUT
   CLK : BOOL;
END_VAR
VAR_OUTPUT
   Q : BOOL;
END_VAR
VAR
   M : BOOL := FALSE;
END_VAR
   Q := CLK AND NOT M;
   M := CLK;
END_FUNCTION_BLOCK
```

只要 CLK 是 FALSE，Q 和 M 就是 FALSE。每次功能调用时，Q 返回 FALSE。当 CLK 检测到上升沿时，Q 返回 TRUE。

✧ 功能块使用举例

变量声明：

```
RTRIGInst : R_TRIG;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
```

POU 语言	程　序
LD	RTRIGInst R_TRIG – CLK Q – –
ST	RTRIGInst(CLK:= VarBOOL1); VarBOOL2 := RTRIGInst.Q;
IL	CAL RTRIGInst(CLK := VarBOOL1) LD RTRIGInst.Q ST VarBOOL2
FBD	RTRIGInst R_TRIG VarBOOL1 – CLK Q — VarBOOL2

6.2.2 F_TRIG——下降沿检测触发器

◆ 功能：用于检测下降沿，功能块源程序定义如下：

```
FUNCTION_BLOCK F_TRIG
VAR_INPUT
  CLK: BOOL;
END_VAR
VAR_OUTPUT
  Q: BOOL;
END_VAR
VAR
  M: BOOL := FALSE;
END_VAR
  Q := NOT CLK AND NOT M;
  M := NOT CLK;
END_FUNCTION_BLOCK
```

只要 CLK 是 TRUE，Q 和 M 保持 FALSE。CLK 是 FALSE，Q 首次返回 TRUE，M 设置为 TRUE。每次功能调用时，Q 返回 FALSE。当 CLK 检测到下降沿时，Q 为 TRUE。

✧ 功能块使用举例

变量声明：

```
FTRIGInst : F_TRIG;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
```

POU 语言	程　序
LD	FTRIGInst F_TRIG CLK　Q
ST	FTRIGInst(CLK:= VarBOOL1); VarBOOL2 := FTRIGInst.Q;
IL	CAL　FTRIGInst(CLK := VarBOOL1) LD　FTRIGInst.Q ST　VarBOOL2
FBD	FTRIGInst F_TRIG VarBOOL1 CLK　Q VarBOOL2

6.3 计数器功能块（Standard.lib）

计数器包括递增计数功能块 CTU、递减计数功能块 CTD 和增减计数功能块 CTUD。

6.3.1 CTU——递增计数器

◆ 功能：递增计数器功能块。

◆ 输入输出说明

✓ CU：BOOL 型，计数输入，当 CU 有从 FALSE 到 TRUE 的上升沿，CV 加 1。

✓ RESET：BOOL 型，RESET 为 TRUE 时，计数变量 CV 初始化为 0。

✓ PV：WORD 型，最大计数值。

✓ Q：BOOL 型，当 CV 大于或等于上限 PV 时，Q 为 TRUE。

✓ CV：WORD 型，当前计数值。

✧ 功能块使用举例

变量声明：

CTUInst : CTU;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL3: BOOL;
VarINT1: INT;
VarINT2: INT;

POU 语言	程　序
LD	CTUInst CTU EN VarBOOL1 – CU　　Q —— VarBOOL3 VarBOOL2 – RESET　　CV —— VarINT2=2 VarINT1=10 – PV
ST	CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTUInst.Q; VarINT2 := CTUInst.CV;
IL	CAL　CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV :=VarINT1) LD　CTUInst.Q ST　VarBOOL3 LD　CTUInst.CV ST　VarINT2
FBD	CTUInst CTU VarBOOL1 – CU　　Q —— VarBOOL3 VarBOOL2 – RESET　　CV —— VarINT2=2 VarINT1=10 – PV

6.3.2 CTD——递减计数器

◆ 功能：递减计数器功能块 CTU。

◆ 输入输出说明

✓ CD：BOOL 型，计数输入，当 CD 有从 FALSE 到 TRUE 的上升沿，若 CV 大于 0，CV 减 1（CV 的值不小于 0）。

✓ LOAD：BOOL 型，LOAD 为 TRUE 时，计数变量 CV 装载为 PV。

✓ PV：WORD 型，装载值。

✓ Q：BOOL 型，当 CV 等于 0 时，Q 为 TRUE。

✓ CV：WORD 型，当前计数值。

✧ 功能块使用举例

变量声明：

CTDInst : CTD;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL3: BOOL;
VarINT1: INT;
VarINT2: INT;

POU 语言	程　序
LD	CTDInst CTD EN VarBOOL1 — CD　　Q — VarBOOL3 VarBOOL2 — LOAD　　CV — VarINT2=8 VarINT1=10 — PV
ST	CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTDInst.Q; VarINT2 := CTDInst.CV;
IL	CAL CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV :=VarINT1) LD CTDInst.Q ST VarBOOL3 LD CTDInst.CV ST VarINT2
FBD	CTDInst CTD VarBOOL1 — CD　　Q — VarBOOL3 VarBOOL2 — LOAD　　CV — VarINT2=8 VarINT1=10 — PV

6.3.3 CTUD——递增递减计数器

- 功能：递增递减计数器功能块 CTUD。
- 输入输出说明
  - ✓ CU：BOOL 型，计数输入，当 CU 有从 FALSE 到 TRUE 的上升沿，CV 加 1。
  - ✓ CD：BOOL 型，计数输入，当 CD 有从 FALSE 到 TRUE 的上升沿，若 CV 大于 0，CV 减 1（CV 的值不小于 0）。
  - ✓ RESET：BOOL 型，RESET 为 TRUE 时，计数变量 CV 初始化为 0。
  - ✓ LOAD：BOOL 型，LOAD 为 TRUE 时，计数变量 CV 装载为 PV。
  - ✓ PV：WORD 型，装载值。
  - ✓ QU：BOOL 型，当 CV 等于 PV 时，QU 为 TRUE。
  - ✓ QD：BOOL 型，当 CV 等于 0 时，QD 为 TRUE。
  - ✓ CV：WORD 型，当前计数值。

✧ 功能块使用举例

变量声明：

```
CTUDInst : CUTD;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
VarBOOL3: BOOL;
VarBOOL3: BOOL;
VarBOOL4: BOOL;
VarBOOL5: BOOL;
VarBOOL6: BOOL;
VarINT1: INT;
VarINT2: INT;
```

<table>
<tr><th>POU 语言</th><th>程　序</th></tr>
<tr><td>LD</td><td>CTUDInst
CTUD
EN
VarBOOL1 — CU　QU — VarBOOL5
VarBOOL2 — CD　QD — VarBOOL6
VarBOOL3 — RESET　CV — VarINT2=2
VarBOOL4 — LOAD
VarINT1=10 — PV</td></tr>
<tr><td>ST</td><td>CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,RESET:=VarBOOL3,
LOAD:=VarBOOL4,PV:=VarINT1)
VarBOOL5 := CTUDInst.QU
VarBOOL6 := CTUDInst.QD
VarINT2 := CTUDInst.CV</td></tr>
<tr><td>IL</td><td>CAL　CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,
RESET:=VarBOOL3, LOAD:=VarBOOL4,PV:= VarINT1)
LD　CTUDInst.QU
ST　VarBOOL5
LD　CTUDInst.QD
ST　VarBOOL6
LD　CTUDInst.CV
ST　VarINT2</td></tr>
<tr><td>FBD</td><td>CTUDInst
CTUD
VarBOOL1 — CU　QU — VarBOOL5
VarBOOL2 — CD　QD — VarBOOL6
VarBOOL3 — RESET　CV — VarINT2=2
VarBOOL4 — LOAD
VarINT1=10 — PV</td></tr>
</table>

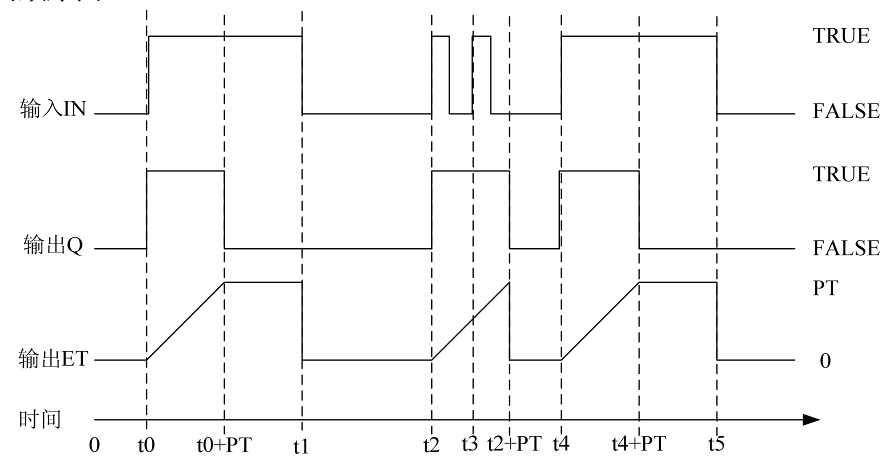
6.4 定时器功能块（Standard.lib）

定时器包括普通定时器功能块 TP、通电延时定时器功能块 TON、断电延时定时器功能块 TOF 和实时时钟 RTC，下面分别描述。

6.4.1 TP——普通定时器

- 功能：普通定时器功能块，函数格式为 TP(IN,PT,Q,ET)。
- 输入输出说明
  - ✓ IN：BOOL 型，定时器初始化信号，当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。
  - ✓ PT：TIME 型，定时时间值。
  - ✓ Q：BOOL 型，若 IN 为 FALSE，则 Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，Q 为 TRUE，在 ET 小于等于 PT 前，IN 无效，ET 等于 PT 时，Q 为 FALSE。
  - ✓ ET：TIME 型，当前时间值，当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。计时完毕后，当 IN 为 FALSE 时，ET 等于 0。

◆ **TP 时间顺序图**



✧ 功能块使用举例

变量声明：

TPInst:TP;
VarBOOL1: BOOL;
VarBOOL2: BOOL;

POU 语言	程　序
LD	TPInst TP EN VarBOOL1 — IN　Q — VarBOOL2 T#5s — PT　ET —
ST	TPInst(IN:= VarBOOL1,PT:= T#5s); VarBOOL2:=TPInst.Q;
IL	CAL　TPInst(IN:= VarBOOL1,PT:=T#5s) LD　TPInst.Q ST　VarBOOL2
FBD	TPInst TP VarBOOL1 — IN　Q — VarBOOL2 T#5s — PT　ET —

6.4.2 TON——通电延时定时器

◆ **功能**：通电延时定时器功能块 TON，函数格式为 TON(IN,PT,Q,ET)。

◆ **输入输出说明**

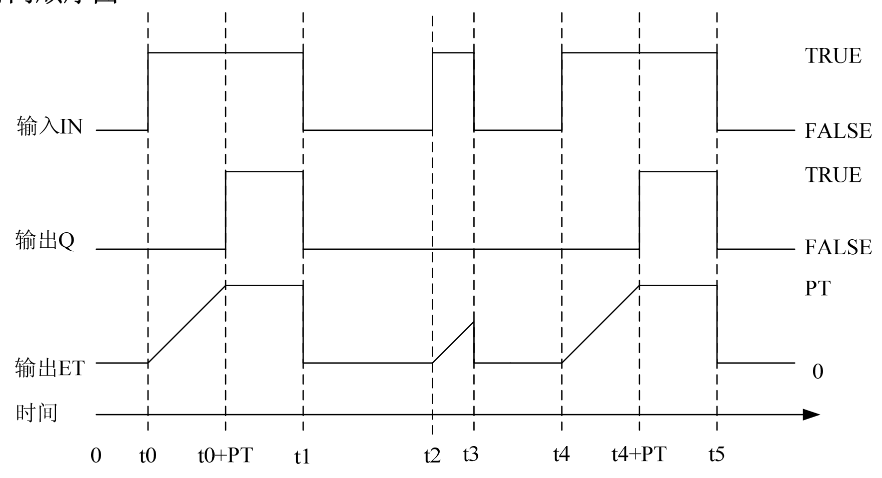
✓ IN：BOOL 型，定时器初始化信号，当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。

✓ PT：TIME 型，定时时间值。

✓ Q：BOOL 型，若 IN 为 FALSE，则 Q 为 FALSE，ET 为 0。IN 为 TRUE，定时器开始工作，ET 等于 PT 时，Q 为 TRUE。

✓ ET：TIME 型，当前时间值，当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 FALSE 时，ET 等于 0。

◆ **TON 时间顺序图**



✧ 功能块使用举例

变量声明：

TONInst :TON;
VarBOOL1: BOOL;
VarBOOL2: BOOL;

POU 语言	程　序
LD	TONInst TON EN VarBOOL1 – IN　Q —— VarBOOL2 T#5s – PT　ET —
ST	TONInst(IN:=VarBOOL1,PT:= T#5s);
IL	CAL　TONInst(IN:= VarBOOL1,PT:=T#5s) LD　TONInst.Q ST　VarBOOL2
FBD	TONInst TON VarBOOL1 – IN　Q —— VarBOOL2 T#5s – PT　ET —

6.4.3 TOF——断电延时定时器

◆ **功能**：断电延时定时器功能块 TOF。

◆ **输入输出说明**

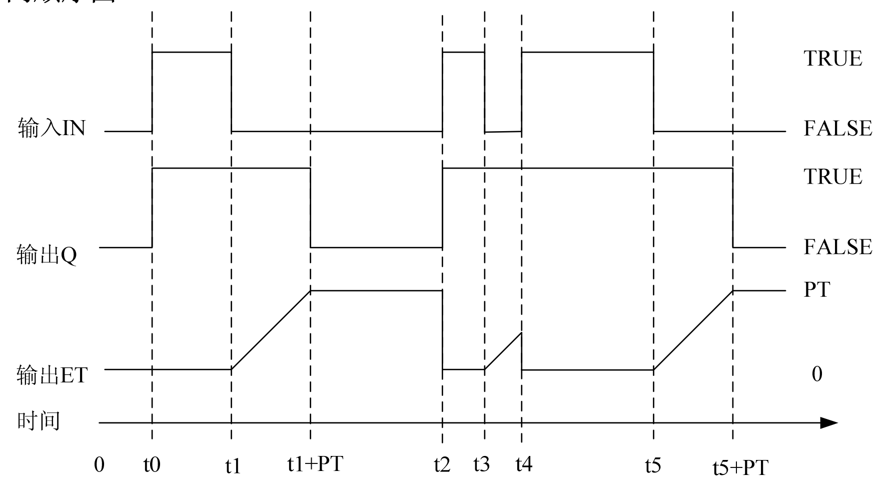
✓ IN：BOOL 型，定时器初始化信号，当 IN 由 TRUE 变成 FALSE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。

✓ PT：TIME 型，输入时间值。

✓ Q：BOOL 型，当 IN 为 FALSE 且 ET 等于 PT 时，Q 为 FALSE。否则，Q 为 TRUE。

✓ ET：TIME 型，当前时间值，当 IN 变成 FALSE 时，ET 以毫秒计数直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 TRUE 时，ET 等于 0，Q 为 TRUE。

◆ **TOF 时间顺序图**



✧ 功能块使用举例

变量声明：

```
TOFInst:TOF;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
```

POU 语言	程　序
LD	TOFInst TOF EN VarBOOL1 — IN　Q — VarBOOL2 T#5s — PT　ET —
ST	TOFInst(IN:= VarBOOL1,PT:=T#5s); VarBOOL2 :=TOFInst.Q;
IL	CAL　TOFInst(IN:= VarBOOL1,PT:= T#5s) LD　TOFInst.Q ST　VarBOOL2
FBD	TOFInst TOF VarBOOL1 — IN　Q — VarBOOL2 T#5s — PT　ET —

6.4.4 RTC——实时时钟

◆ 功能：在给定时间启动，返回当前日期和时间。函数格式为 RTC(EN,PDT,Q,CDT)。

◆ 输入输出说明

✓ EN：BOOL 型，启动该功能块。

✓ PDT：DT 型，时间基值。

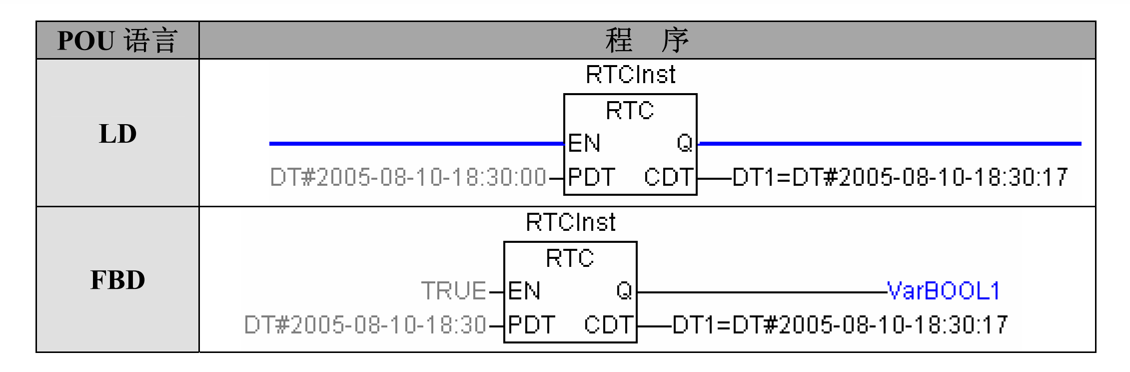
✓ Q：BOOL 型，EN 为 FALSE 时，Q 为 FALSE，EN 为 TRUE 时，Q 为 TRUE。

✓ CDT：DT 型，EN 为 FALSE 时，CDT 为 1970-01-01-00:00:00，EN 为 TRUE 时，CDT 从 PDT 中的时间开始计时。

✧ 功能块使用举例

变量声明：

```
RTCInst: RTC;
DT1: DT;
VarBOOL1: BOOL;
```



POU 语言	程　序
LD	RTCInst RTC EN Q DT#2005-08-10-18:30:00-PDT CDT-DT1=DT#2005-08-10-18:30:17
FBD	RTCInst RTC TRUE-EN Q-VarBOOL1 DT#2005-08-10-18:30-PDT CDT-DT1=DT#2005-08-10-18:30:17

6.5 模拟量模块处理功能块（Hollysys_PLC_Analog.lib）

6.5.1 Analog_IN——模拟量输入功能块

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	ANALOG_IN EN Q Address

◆ 输入输出说明

输入数据存放地址	含　义		
%IWxx	模拟量输入通道依次对应 PLC 配置中所规定的 IW 区地址		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效，不对模拟输入模块进行扫描 1：使能，对模拟输入模块进行扫描
Address	该模块节点 id	BYTE	0—7（与 PLC 配置里的节点 id 一致）
输出	**功能**	**数据类型**	**值**
Q	是否读到有效数据	BOOL	0：未读到数据 1：读到有效数据

✧ **Analog_IN 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: Analog_IN;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
        EN          Example
                    Analog_IN                               T_OR_F
   ----| |--------EN        Q------------------------------( )---
                0-Address
```

图 6-5-1

图 6-5-1 中 Address 处的输入值应与图 6-5-2 中 PLC 配置表中鼠标处的节点 id 一致。

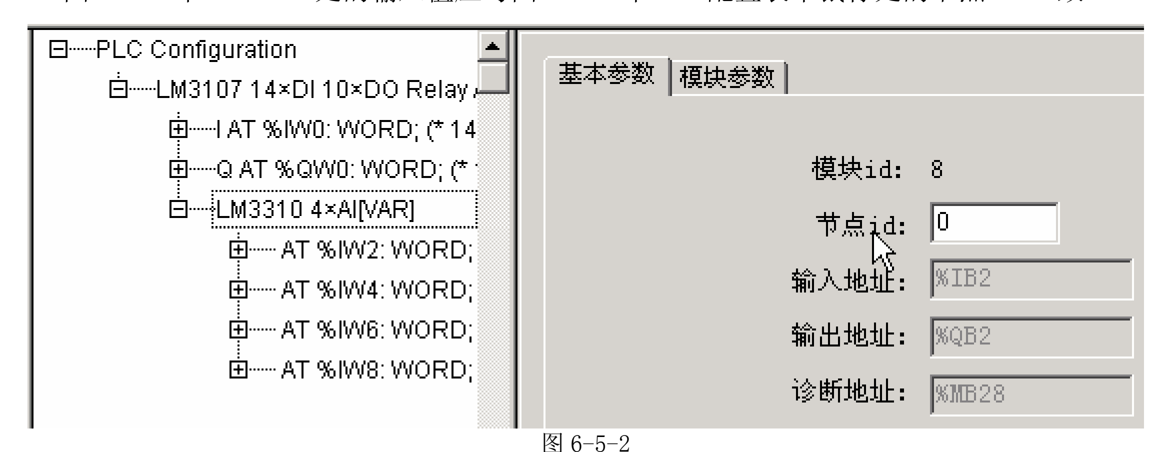


图 6-5-2

梯形图描述:

EN 置位时，对模拟量输入模块进行扫描，4 个通道的扫描值分别存储在上图所示的%IW2、%IW4、%IW6、%IW8。

EN 复位时，不对模拟量输入模块进行扫描。

注意

➢ 如果要使用多个模拟量输入模块，则需要配置多个 Analog_IN 功能块，每个功能块对应的 Address 与对应模块的节点 id 保持一致。

6.5.2 Analog_OUT——模拟量输出功能块

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	ANALOG_OUT EN Q Address

◆ 输入输出说明

数据输出存放地址	含 义		
%QWxx	PLC 配置中所规定的 QW 区地址依次对应模拟量输出模块的通道		
输入	功能	数据类型	值
EN	使能	BOOL	0：无效，不对模拟输出模块进行扫描 1：使能，对模拟输出模块进行扫描
Address	该模块节点 id	BYTE	0—7（与 PLC 配置里的节点 id 一致）
输出	功能	数据类型	值
Q	是否正确输出数据	BOOL	0：未输出数据 1：正确输出数据

✧ **Analog_OUT 功能块举例（变量定义与梯形图）**

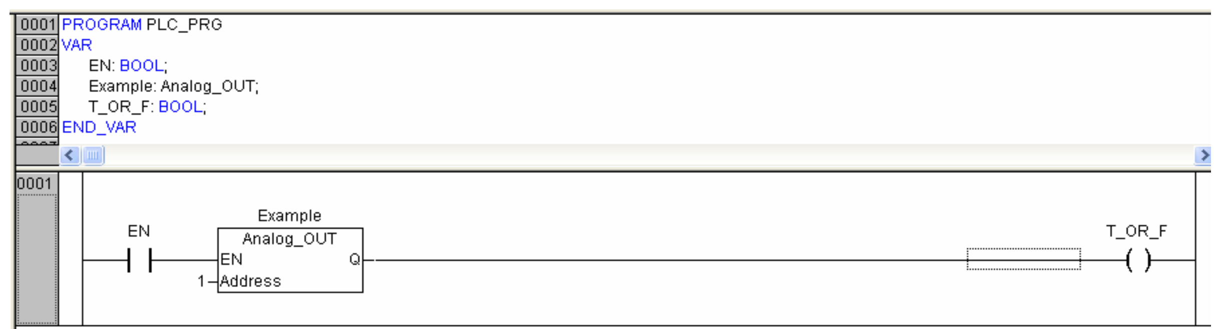


图 6-5-3

图 6-5-3 中 Address 处的输入值应与图 6-5-4 中 PLC 配置表中鼠标处的节点 id 一致。

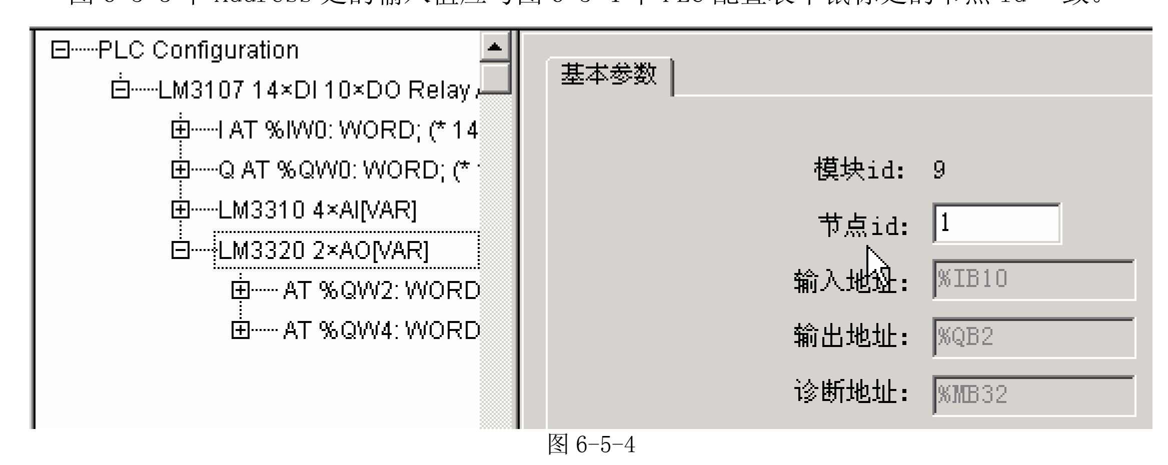


图 6-5-4

梯形图描述:

EN 置位时，对模拟量输出模块进行扫描，2 个通道的输出值分别存储在上图所示的%QW2、%QW4。

EN 复位时，不对模拟量输出模块进行扫描。

注意

➢ 如果要使用多个模拟量输出模块，则需要配置多个 Analog_OUT 功能块，每个功能块对应的 Address 与对应模块的节点 id 保持一致。

6.6 RS232 自由口通讯功能块（Hollysys_PLC_Comm.lib）

6.6.1 Set_COMM_PRMT——RS232 自由口通讯参数设置

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	SET_COMM_PRMT EN　Q Mode

◆ 输入输出说明

<table>
<tr><th>输入</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td colspan="8">0：无效
1：上升沿使能</td></tr>
<tr><td rowspan="10">Mode</td><td rowspan="10">通讯模式配置
（不支持 7 位无校验，这时默认为 7 位偶校验）</td><td rowspan="10">BYTE</td><td colspan="8">Bits</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">校验</td><td>位数</td><td colspan="3">波特率</td><td colspan="2">未定义</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">校验设置</td></tr>
<tr><td colspan="8">保留现所有通讯参数：0 0（启动自由口）
偶校验：　　0 1
无校验：　　1 0
奇校验：　　1 1</td></tr>
<tr><td colspan="8">位数设置</td></tr>
<tr><td colspan="8">8 位：0
7 位：1</td></tr>
<tr><td colspan="8">波特率设置</td></tr>
<tr><td colspan="8">38400bps：0 0 0
19200bps：0 0 1
9600bps：　0 1 0
4800bps：　0 1 1
2400bps：　1 0 0
1200bps：　1 0 1
600bps：　 1 1 0
300bps：　 1 1 1</td></tr>
<tr><th>输出</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>Q</td><td>设置完成标志</td><td>BOOL</td><td colspan="8">0：设置未完成
1：设置已完成</td></tr>
</table>

注意

➢ 使用 Set_COMM_PRMT 功能块设定自由口参数后，要想恢复原编程系统下装/调试功能，需将 RUN/STOP 开关拨到 STOP 位置，才可进行编程系统登录。

✧ **Set_COMM_PRMT 功能块举例（变量定义与梯形图）**

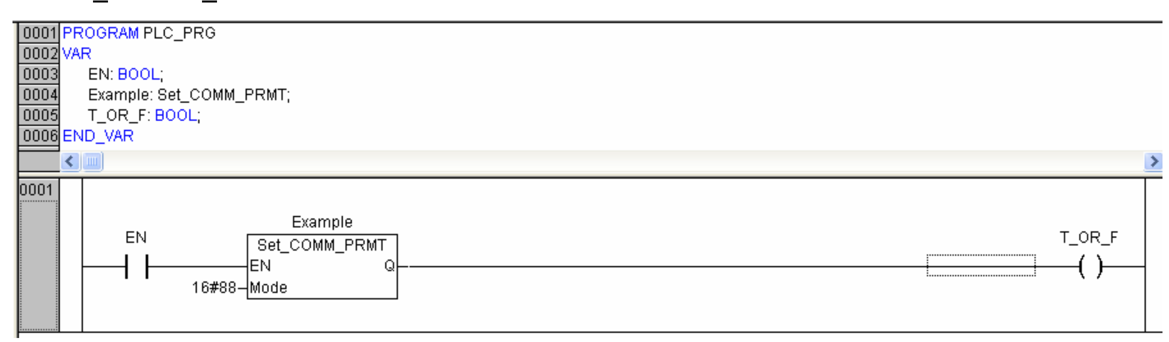


图 6-6-1

如下表，Mode 十六进制为 88（二进制为 10 0 010 00），则自由口为无校验、8 位、波特率 9600bps。

Mode=16#88							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	1	0	0	0

梯形图描述:

EN 置位并保持时（上升沿），则 RS232 口设置成无校验、8 位、波特率 9600bps 的自由口。

EN 复位时，保持原设置不变。

6.6.2 COMM_SEND——RS232 自由口通讯数据发送

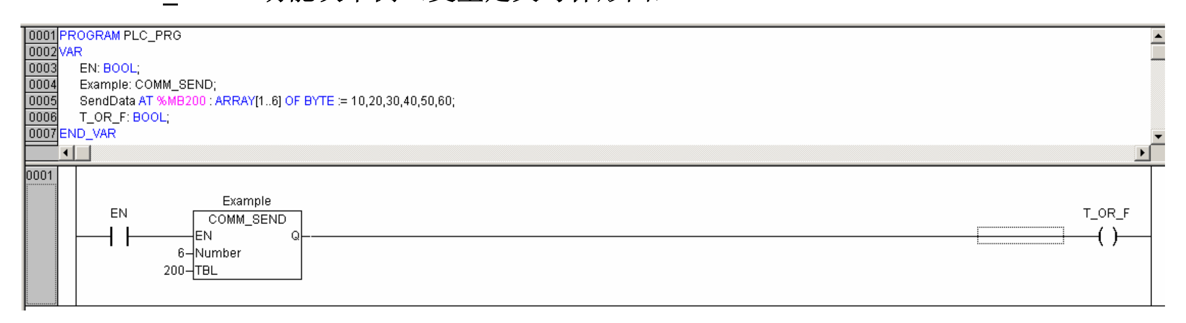
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	COMM_SEND EN Q Number TBL

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Number	发送字节数	BYTE	
TBL	M 区发送缓冲区首地址	INT	
输出	**功能**	**数据类型**	**值**
Q	发送完成标志	BOOL	0：发送未完成 1：发送已完成

✧ **COMM_SEND 功能块举例（变量定义与梯形图）**


```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: COMM_SEND;
0005     SendData AT %MB200 : ARRAY[1..6] OF BYTE := 10,20,30,40,50,60;
0006     T_OR_F: BOOL;
0007 END_VAR
```


梯形图描述:

在 RS232 自由口模式下，EN 置位并保持时（上升沿），将 SendData 数组中的 6 个字节发送出去，Q 等于 1，保持。

EN 复位时，不进行发送。

6.6.3 COMM_RECEIVE——RS232 自由口通讯数据接收

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	COMM_RECEIVE EN　Q Number TBL

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Number	接收字节数	BYTE	
TBL	M 区接收缓冲区首地址	INT	
输出	**功能**	**数据类型**	**值**
Q	接收完成标志	BOOL	0：接收未完成 1：接收已完成

✧ **COMM_ RECEIVE** 功能块举例（变量定义与梯形图）

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: COMM_RECEIVE;
0005     RecData AT %MB200: ARRAY[1..6] OF BYTE;
0006 END_VAR
```


梯形图描述:

在 RS232 自由口模式下，上电 EN 等于 0，当开始运行时第一个扫描周期接收 6 个字符保存到 RecData 数组中，Q 等于 1，从而 EN 等于 1。第二个扫描周期时该功能块不进行接收，Q 等于 0，从而 EN 等于 0。第三个扫描周期再次接受 6 个字符保存到 RecData 数组中，如此循环接收数据。

6.6.4 Reset_COMM_PRMT——RS232 恢复 G3 专有协议

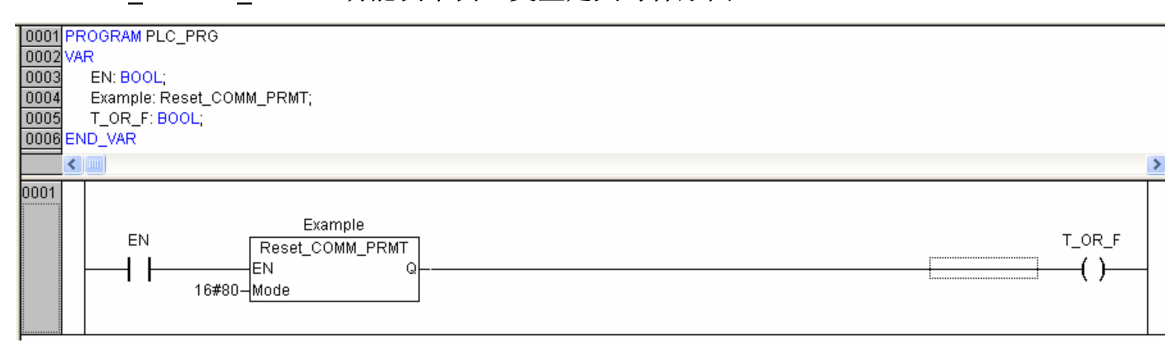
◆ 功能块示例

POU 语言	程　序
梯形图（LD）	RESET_COMM_PRMT EN　Q Mode

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Mode	通讯模式配置 （不支持 7 位无校验，这时默认为 7 位偶校验）	BYTE	Bits 7 6：校验；5：位数；4 3 2：波特率；1 0：未定义 校验设置 保留现所有通讯参数：0 0 （恢复 G3 专有协议） 偶校验：0 1 无校验：1 0 奇校验：1 1 位数设置 8 位：0 7 位：1 波特率设置 38400bps：0 0 0 19200bps：0 0 1 9600bps：0 1 0 4800bps：0 1 1 2400bps：1 0 0 1200bps：1 0 1 600bps：1 1 0 300bps：1 1 1
输出	**功能**	**数据类型**	**值**
Q	设置完成标志	BOOL	0：设置未完成 1：设置已完成

✧ **Reset_COMM_PRMT 功能块举例（变量定义与梯形图）**



如下表，Mode 十六进制为 80（二进制为 10 0 000 00），则 G3 专有协议为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

梯形图描述:

EN 置位并保持时（上升沿），将 RS232 口设置成无校验、8 位、波特率 38400bps 的 G3 专有协议模式。

6.7 RS485 自由口通讯功能块（Hollysys_PLC_Comm2.lib）

6.7.1 Set_COMM2_PRMT——RS485 自由口通讯参数设置

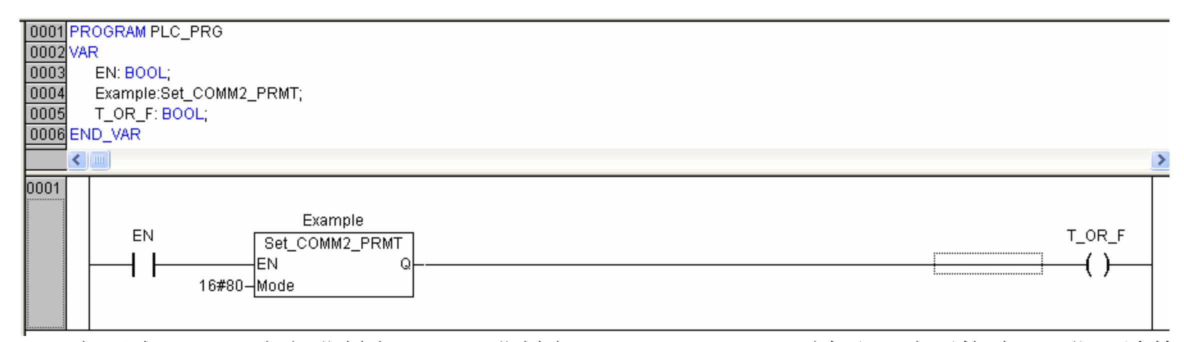
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	SET_COMM2_PRMT EN Q Mode

◆ 输入输出说明

<table>
<tr><th>输入</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td colspan="8">0：无效
1：上升沿使能</td></tr>
<tr><td rowspan="11">Mode</td><td rowspan="11">通讯模式配置</td><td rowspan="11">BYTE</td><td colspan="8">Bits</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">校验</td><td>位数</td><td colspan="3">波特率</td><td colspan="2">未定义</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">校验设置</td></tr>
<tr><td colspan="8">保留现所有通讯参数：0 0 （启动自由口）
偶校验： 0 1
无校验： 1 0
奇校验： 1 1</td></tr>
<tr><td colspan="8">位数设置</td></tr>
<tr><td colspan="8">8 位：0
7 位：1</td></tr>
<tr><td colspan="8">波特率设置</td></tr>
<tr><td colspan="8">38400bps：0 0 0
19200bps：0 0 1
9600bps： 0 1 0
4800bps： 0 1 1
2400bps： 1 0 0
1200bps： 1 0 1
600bps： 1 1 0
300bps： 1 1 1</td></tr>
<tr><td colspan="8"></td></tr>
<tr><th>输出</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>Q</td><td>设置完成标志</td><td>BOOL</td><td colspan="8">0：设置未完成
1：设置已完成</td></tr>
</table>

✧ **Set_COMM2_PRMT 功能块举例（变量定义与梯形图）**



如下表，Mode 十六进制为 80（二进制为 10 0 000 00），则自由口为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

梯形图描述:

EN 置位并保持时（上升沿），将 RS485 口设置成无校验、8 位、波特率 38400bps 的自由口。
EN 复位时，保持原设置不变。

6.7.2 COMM2_SEND——RS485 自由口通讯数据发送

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	COMM2_SEND EN　Q Number TBL

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Number	发送字节数	BYTE	
TBL	M 区发送缓冲区首地址	INT	
输出	**功能**	**数据类型**	**值**
Q	发送完成标志	BOOL	0：发送未完成 1：发送已完成

✧ **COMM2_SEND** 功能块举例（变量定义与梯形图）

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: COMM2_SEND;
0005     SendData AT %MB200 : ARRAY[1..6] OF BYTE := 10,20,30,40,50,60;
0006     T_OR_F: BOOL;
0007 END_VAR
```

```
0001
      EN        Example                                   T_OR_F
               COMM2_SEND
  ---| |------|EN       Q|-------------------------------( )--
            6-|Number    |
          200-|TBL       |
```

梯形图描述:

在 RS485 自由口模式下，EN 置位并保持时（上升沿），将 SendData 数组中的 6 个字节发送出去，Q 等于 1，保持。

EN 复位时，不进行发送。

6.7.3 COMM2_RECEIVE——RS485 自由口通讯数据接收

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	COMM2_RECEIVE EN　Q Number TBL

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Number	接收字节数	BYTE	
TBL	M 区接收缓冲区首地址	INT	
输出	**功能**	**数据类型**	**值**
Q	接收完成标志	BOOL	0：接收未完成 1：接收已完成

✧ **COMM2_RECEIVE** 功能块举例（变量定义与梯形图）

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: COMM2_RECEIVE;
0005     RecData AT %MB200: ARRAY[1..6] OF BYTE;
0006 END_VAR
```

```
0001
      EN        Example                                   EN
              COMM2_RECEIVE
  ---|/|------|EN       Q|-------------------------------( )--
            6-|Number    |
          200-|TBL       |
```

梯形图描述:

在 RS485 自由口模式下，上电 EN 等于 0，当开始运行时第一个扫描周期接收 6 个字符保存到 RecData 数组中，Q 等于 1，从而 EN 等于 1。第二个扫描周期时该功能块不进行接收，Q 等于 0，从而 EN 等于 0。第三个扫描周期再次接受 6 个字符保存到 RecData 数组中，如此循环接收数据。

6.7.4 Reset_COMM2_PRMT——RS485 恢复 G3 专有协议

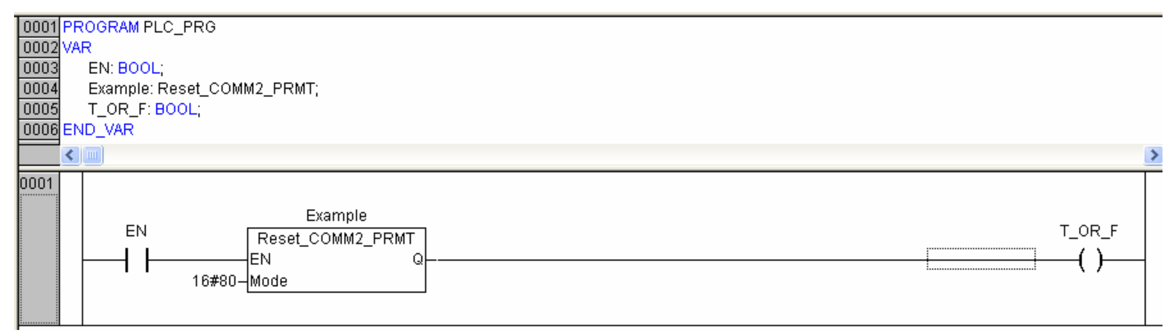
◆ 功能块示例

POU 语言	程　序
梯形图（LD）	RESET_COMM2_PRMT EN　Q Mode

◆ 输入输出说明

<table>
<tr><th>输入</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td colspan="8">0：无效
1：上升沿使能</td></tr>
<tr><td rowspan="12">Mode</td><td rowspan="12">通讯模式配置</td><td rowspan="12">BYTE</td><td colspan="8">Bits</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">校验</td><td>位数</td><td colspan="3">波特率</td><td colspan="2">未定义</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">校验设置</td></tr>
<tr><td colspan="8">保留现所有通讯参数：0 0
（恢复 G3 专有协议）
偶校验：　0 1
无校验：　1 0
奇校验：　1 1</td></tr>
<tr><td colspan="8">位数设置</td></tr>
<tr><td colspan="8">8 位：0
7 位：1</td></tr>
<tr><td colspan="8">波特率设置</td></tr>
<tr><td colspan="8">38400bps：0 0 0
19200bps：0 0 1
9600bps：　0 1 0
4800bps：　0 1 1
2400bps：　1 0 0
1200bps：　1 0 1
600bps：　1 1 0
300bps：　1 1 1</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8"></td></tr>
<tr><th>输出</th><th>功能</th><th>数据类型</th><th colspan="8">值</th></tr>
<tr><td>Q</td><td>设置完成标志</td><td>BOOL</td><td colspan="8">0：设置未完成
1：设置已完成</td></tr>
</table>

✧ **Reset_COMM2_PRMT 功能块举例（变量定义与梯形图）**



如下表，Mode 十六进制为 80（二进制为 10 0 000 00），则 G3 专有协议为无校验、8 位、波特率 38400bps。

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
校验		位数	波特率			未定义	
1	0	0	0	0	0	0	0

梯形图描述:

EN 置位并保持时（上升沿），将 RS485 口设置成无校验、8 位、波特率 38400bps 的 G3 专有协议模式。

6.8 Profibus-DP 功能块（Hollysys_PLC_DPSlave.lib）

6.8.1 DP_Slave——Profibus-DP 从站功能块（LM3401）

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	DP_SLAVE EN Q Address Err

◆ 输入输出说明

CPU 模块输入输出数据存放地址			
%IWxx %QWxx	PLC 配置中规定的 IW 和 QW 区地址		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效，不对 DP 模块进行扫描 1：使能，对 DP 模块进行扫描
Address	该模块节点 id	BYTE	0—7(与 PLC 配置里的节点 id 一致)
输出	**功能**	**数据类型**	**值**
Q	是否读到有效数据	BOOL	0：未读到数据 1：读到有效数据

Err	与从站模块通讯故障号	BYTE	0：无故障 1：等待 DP 从站模块 Ready 错 2：读中断 DP 从站模块错 3：读应答错 4：读数据长度错 5：读数据错 6：写数据错 7：写中断 DP 从站模块错

✧ **DP_Slave 功能块举例（变量定义与梯形图）**

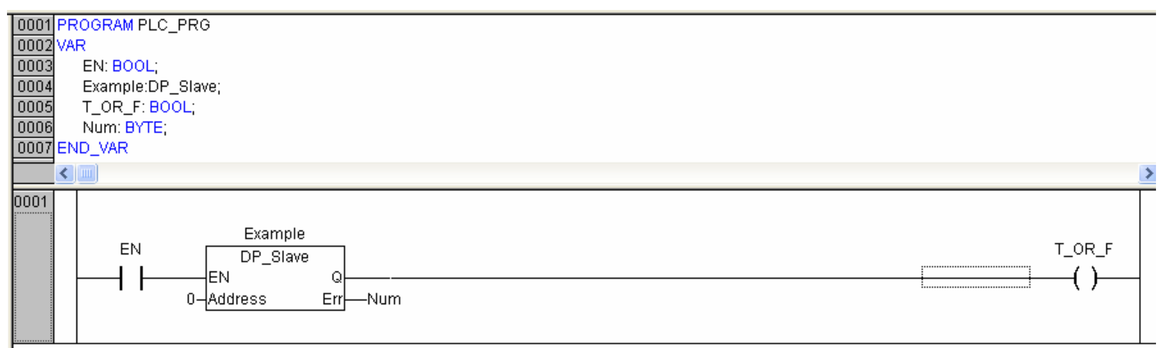


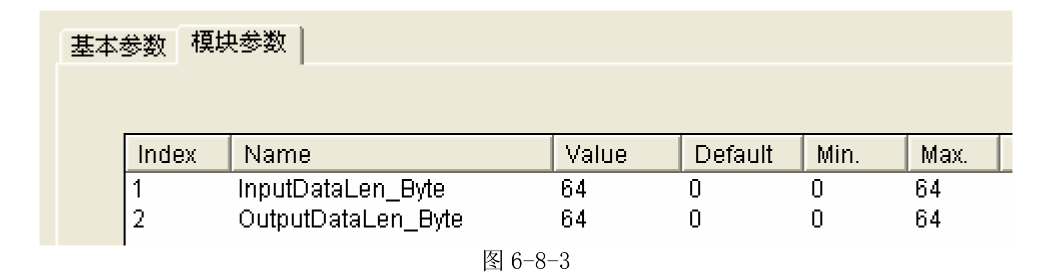
图 6-8-1

图 6-8-1 中 Address 处的输入值应与图 6-8-2 中 PLC 配置表中鼠标处的节点 id 一致。



图 6-8-2

在图 6-8-3 中模块参数的 Value 处配置输入输出区的大小。


基本参数　模块参数

Index	Name	Value	Default	Min.	Max.
1	InputDataLen_Byte	64	0	0	64
2	OutputDataLen_Byte	64	0	0	64

图 6-8-3

梯形图描述：

EN 置位时，对 DP 模块进行扫描。

EN 复位时，不对 DP 模块进行扫描。

注意

➢ 关于 DP 模块的使用请参考第九章例程。

6.9 以太网功能块（Hollysys_PLC_EtherNet.lib）

6.9.1 EtherNet_TCP——以太网功能块（LM3403）

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	ETHERNET_TCP EN Q Address Err

◆ 输入输出说明

CPU 模块输入输出数据存放地址			
%IWxx %QWxx	PLC 配置中规定的 IW 和 QW 区地址		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效，不对以太网模块扫描 1：使能，对以太网模块扫描
Address	该模块节点 id	BYTE	0—7（与 PLC 配置里的节点 id 一致）
输出	**功能**	**数据类型**	**值**
Q	是否读到有效数据	BOOL	0：未读到数据 1：读到有效数据
Err	与以太网模块 通讯故障号	BYTE	0：无故障 1：等待以太网模块 Ready 错 2：读中断以太网模块错 3：读应答错 4：读数据长度错 5：读数据错 6：写数据错 7：写中断以太网模块错

✧ **EtherNet_TCP 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: EtherNet_TCP;
0005     T_OR_F: BOOL;
0006     Err: BYTE;
0007 END_VAR
```

0001: EN — Example (EtherNet_TCP: EN, Q; 0—Address, Err—Err) — T_OR_F ( )

图 6-9-1

图 6-9-1 中 Address 处的输入值应与图 6-9-2 中 PLC 配置表中鼠标处的节点 id 一致。

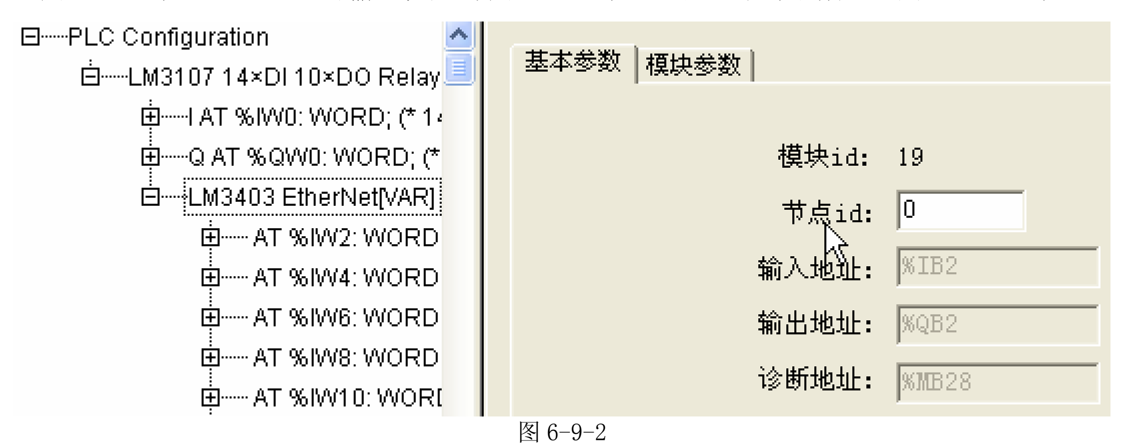
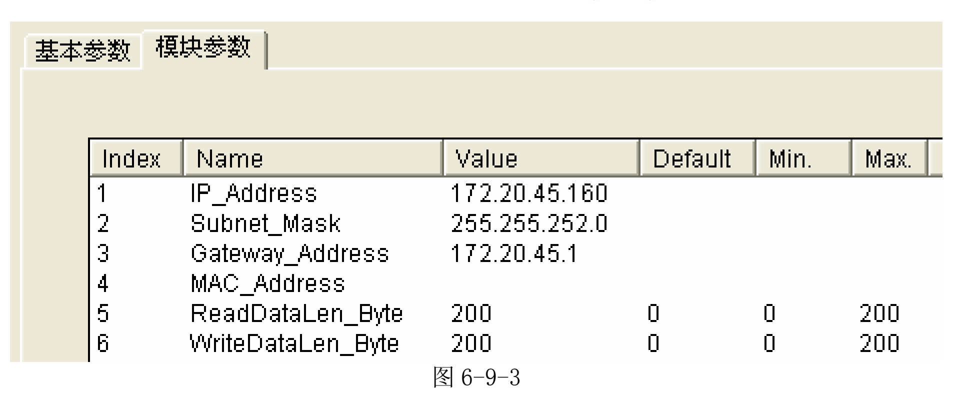


图 6-9-2

在图 6-9-3 中分别配置 IP 地址、子网掩码、网关、输入输出区的大小。



Index	Name	Value	Default	Min.	Max.
1	IP_Address	172.20.45.160			
2	Subnet_Mask	255.255.252.0			
3	Gateway_Address	172.20.45.1			
4	MAC_Address				
5	ReadDataLen_Byte	200	0	0	200
6	WriteDataLen_Byte	200	0	0	200

图 6-9-3

梯形图描述：

EN 置位时，对以太网模块进行扫描。

EN 复位时，不对以太网模块进行扫描。

注意

- 关于以太网模块的使用请参考附录 B 例程。

6.10 硬件实时时钟功能块（Hollysys_PLC_HDRTC.lib）

6.10.1 Set_HD_RTC——设置实时时钟（DT 数据格式）

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	SET_HD_RTC EN Q PDT DAY

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
PDT	日期/时间	DT	DT#1970-01-01-00:00:00 至 DT#2069-12-31-23:59:59
Day	星期值	BYTE	1—7
输出	**功能**	**数据类型**	**值**
Q	是否设定完成	BOOL	0：未设定完成 1：设定完成

✧ **Set_HD_RTC 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: Set_HD_RTC;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
      EN              Example
                    Set_HD_RTC
     -| |----------|EN        Q|---------------------------------T_OR_F ( )
DT#2005-08-08-8:58:00-|PDT      |
                   1-|DAY      |
```

梯形图描述:

EN 置位时，如图所示的日期、时间、星期将被设置到 PLC 硬件实时时钟之中，当前的实时时钟是：2005-08-08-8:58:00，星期一。

注意

➢ 设置日期时间时不可以超出规定范围。

6.10.2 Set_HD_RTC_X——设置实时时钟（普通数据格式）

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	SET_HD_RTC_X: EN, nYear, nMonth, nDate, nHour, nMinute, nSecond, DAY → Q

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
nYear	年	INT	1970-2069
nMonth	月	BYTE	1-12
nDate	日	BYTE	1-31
nHour	时	BYTE	0-24
nMinute	分	BYTE	0-60
nSecond	秒	BYTE	0-60
DAY	星期	BYTE	1-7
输出	**功能**	**数据类型**	**值**
Q	是否设定完成	BOOL	0：未设定完成 1：设定完成

✧ **Set_HD_RTC_X 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: Set_HD_RTC_X;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
        EN        Example
                  Set_HD_RTC_X                                        T_OR_F
   ---| |--------|EN          Q|-----------------------------------------( )---
            2005-|nYear        |
               8-|nMonth       |
              18-|nDate        |
              13-|nHour        |
              10-|nMinute      |
              15-|nSecond      |
               4-|DAY          |
```

梯形图描述：

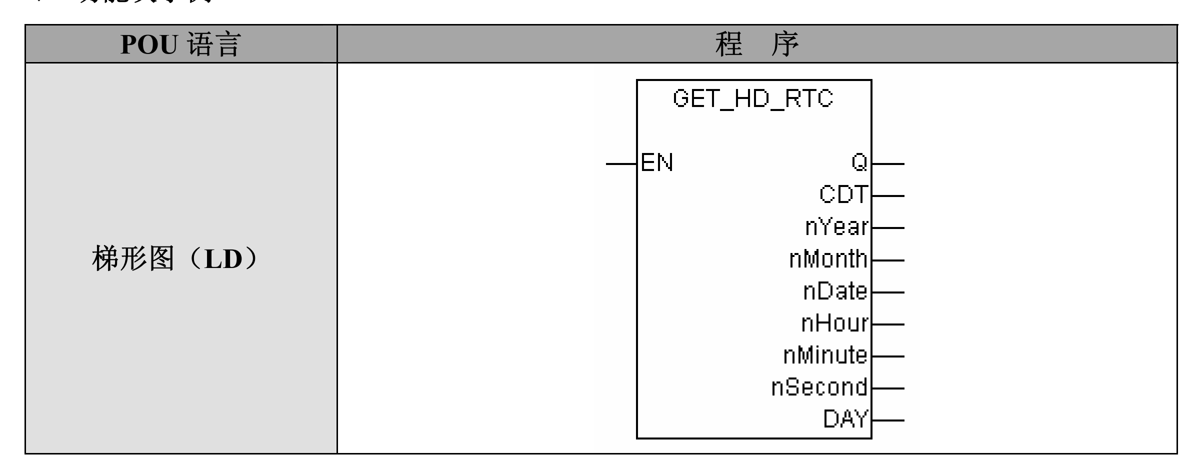
EN 置位时，如图所示的日期、时间、星期将被设置到 PLC 硬件实时时钟之中，当前的实时时钟是 2005 年 8 月 18 日 13 点 10 分 15 秒，星期四。

注意

➢ 设置日期时间时不可以超出规定范围。

6.10.3 Get_HD_RTC——读取实时时钟日期/时间/星期

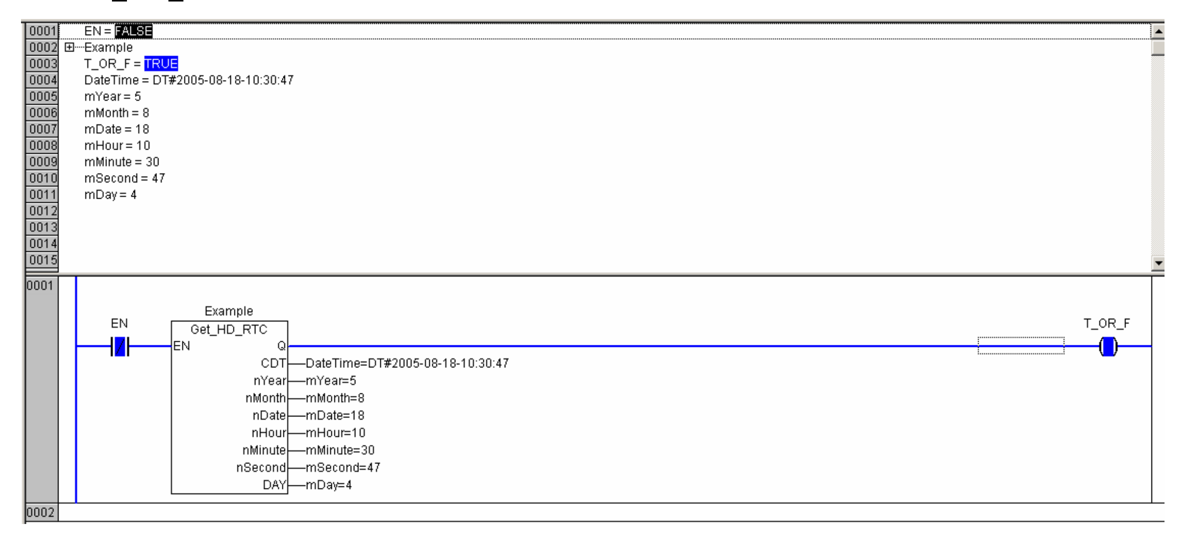
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
输出	**功能**	**数据类型**	**值**
Q	是否读出数据	BOOL	0：未读出数据 1：正确取出数据
CDT	日期/时间	DT	DT#1970-01-01-00:00:00 至 DT#2069-12-31-23:59:59
nYear	年	INT	0-99（年份后两位）
nMonth	月	BYTE	1-12
nDate	日	BYTE	1-31
nHour	时	BYTE	0-24
nMinute	分	BYTE	0-60
nSecond	秒	BYTE	0-60
DAY	星期	BYTE	1-7

✧ **Get_HD_RTC** 功能块举例（运行中的梯形图）



梯形图描述:

EN 置位时，CDT 以 DT 型数据格式输出 PLC 实时时钟的日期、时间，nYear 以 INT 型数据格式输出年，nMonth、nDate、nHour、nMinute、nSecond、DAY 分别以 BYTE 型数据格式输出月、日、时、分、秒、星期，Q 等于 1，当前的实时时钟是：2005-08-18-10:30:47，星期四。

EN 复位时，Q 等于 0，CDT、nYear、nMonth、nDate、nHour、nMinute、nSecond、DAY 保持最后一次输出值不变。

6.11 实时时钟报警功能块（Hollysys_PLC_HDRTCALM.lib）

6.11.1 Get_HDRTC_ALM——读取硬件实时时钟报警时间/星期

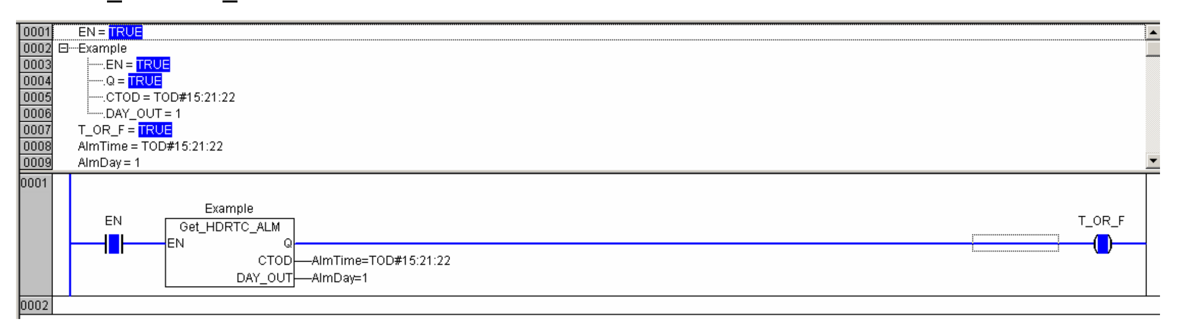
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	GET_HDRTC_ALM EN Q CTOD DAY_OUT

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
输出	**功能**	**数据类型**	**值**
Q	是否取出报警数据	BOOL	0：未读出报警数据 1：正确取出报警数据
CTOD	当前报警时间	TOD	
DAY_OUT	当前报警星期值	BYTE	1—7

✧ **Get_HDRTC_ALM 功能块举例（运行中的梯形图）**



梯形图描述:

EN 置位时，CDOT 输出 PLC 的实时时钟报警时间，DAY_OUT 输出 PLC 的实时时钟报警星期，Q 等于 1，当前的报警时间是：15:21:22，星期是：星期一。

EN 复位时，Q 等于 0，CTOD 和 DAY_OUT 保持最后一次输出值不变。

6.11.2 Set_HDRTC_ALM——设置硬件实时时钟报警时间/星期

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	SET_HDRTC_ALM EN　Q PTOD DAY_IN MODE

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
PTOD	时间	TOD	
DAY_IN	星期值	BYTE	1—7
MODE	报警模式	BYTE	0：触发报警事件关闭 1：当星期、小时、分钟、秒匹配时，触发报警事件（每星期触发一次） 2：当小时、分钟、秒匹配时，触发报警事件（每天触发一次） 3：当分钟、秒匹配时，触发报警事件（每小时触发一次） 4：当秒匹配时，触发报警事件（每分钟触发一次） 5：每秒触发报警事件一次
输出	**功能**	**数据类型**	**值**
Q	是否正确地设定报警数据	BOOL	0：未设定报警数据 1：正确设定报警数据

◆ 报警关联事件

HD_RTC_ALM 0 interrupt

注意

➢ 事件使用方法参见附录 B。

✧ **Set_HDRTC_ALM 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: Set_HDRTC_ALM;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
        EN                 Example
                        Set_HDRTC_ALM                                T_OR_F
   ----| |-------------|EN           Q|-------------------------------( )----
        TOD#18:19:20---|PTOD          |
                   3---|DAY_IN        |
                   1---|MODE          |
```

梯形图描述:

EN 置位时，如图所示设置 PLC 的实时报警时间/星期，Q 等于 1，当前设置的报警时间星期是：18:19:20，星期三。因为选择模式 1，所以每个星期三的 18 点 19 分 20 秒触发一次报警。

EN 复位时，Q 等于 0。

注意

➢ 每次报警事件触发后应重新写入或读取一次报警数据，否则不能启动下一次报警。

6.12 自设定组脉冲发送功能块（Hollysys_PLC_PTCYC.lib）

6.12.1 PTO_CYC——自设定组脉冲循环发送

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	PTO_CYC EN Q Number TBL CYCNum

◆ 输入输出说明

外部端子输出	功能		
Q1.1	高速脉冲输出		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效 1：上升沿使能
Number	脉冲组数（脉冲频率+发送个数为 1 组）	BYTE	1-255
TBL	自设定脉冲频率与个数表的 M 区相对首地址	INT	
CYCNum	循环的次数	WORD	0：无限循环 1-65535：循环次数
输出	**功能**	**数据类型**	**值**
Q	设置完成标志	BOOL	0：设置未完成 1：设置已完成

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
PTO_CYC	PTO_PWM0	T12	不可使用功能块 B
	PTO_PWM0_Run		

◆ 自设定脉冲频率、个数方法

脉冲频率与对应脉冲个数定义方式：

✓ 定义为 DWORD 型数组。

✓ 单独定义每组里的每个变量，需要在连续地址上定义（DWORD）。

M 区数组定义举例

```
PTOData AT %MD200 : ARRAY [1..6] OF DWORD := 1000，500，2000，1200，1500，800;
```

✧ PTO_CYC 功能块举例（变量定义与梯形图）

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: PTO_CYC;
0005     PTOData AT %MD200 : ARRAY[1..6] OF DWORD := 1000,500,2000,1200,1500,800;
0006     T_OR_F: BOOL;
0007 END_VAR
```

```
0001
        EN          Example
                    PTO_CYC                                              T_OR_F
   ---| |--------|EN         Q|-------------------------------------------( )---
               3-|Number      |
             200-|TBL         |
               6-|CYCNum      |
```

梯形图描述:

EN 置位时，Q1.1 开始发送脉冲，按照如下格式不间断发送（%MD200 处开始定义的数组）。

1kHz 发送 500 个脉冲——2KHz 发送 1200 个脉冲——1.5KHz 发送 800 个脉冲。

CYCNum 等于 6，所以共完成 6 个上述循环，中间不间断。

EN 复位时，Q1.1 停止发送脉冲。

第7章 PowerPro 外部扩展功能块

7.1 Modubs 功能块（Hollysys_PLC_Ex.lib）

7.1.1 SET_LOCAL_ADDRESS——设置本机 Modbus 从站通讯地址

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	SET_LOCAL_ADDRESS EN Q Address

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上电/下装后首次置位有效
Address	Modbus 从站地址	BYTE	1—247：从站号
输出	**功能**	**数据类型**	**值**
Q	是否设置完毕	BOOL	0：EN 复位 1：设置完毕，保持

✧ **SET_LOCAL_ADDRESS 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Example: Set_Local_Address;
0004     EN: BOOL;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
      EN          Example
                  Set_Local_Address                          T_OR_F
 ----| |---------|EN              Q|-------------------------( )--
              51-|Address          |
```

梯形图描述:

上电或者下装后，EN 首次置位时，该功能块使能，从站设置完毕后，Q 等于 1。
EN 复位时，Q 等于 0，但从站地址为先前设定值，不会改变。
此时默认通讯参数为 38400bps、8 位、无校验。

注意

➢ 如果所需通讯参数不是 38400bps、8 位、无校验，则需要使用 Reset_COMM_PRMT 功能块设置相应的通讯参数，然后再使用 SET_LOCAL_ADDRESS 功能块设置站地址，程序下装前需要将 RUN/STOP 开关拨到 STOP 位置，程序下装后再将 RUN/STOP 开关拨到 RUN 位置运行程序。

7.1.2 GET_LOCAL_ADDRESS——读取本机 Modbus 从站通讯地址

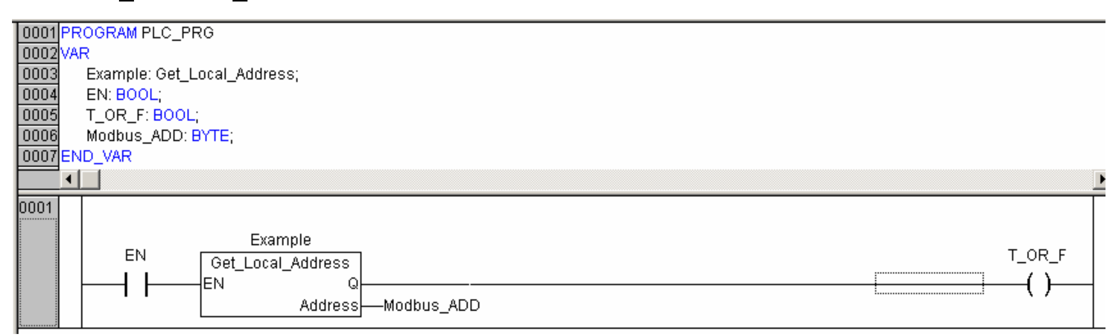
◆ 功能块示例

POU 语言	程　序
梯形图（LD）	GET_LOCAL_ADDRESS EN　Q Address

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
输出	**功能**	**数据类型**	**值**
Q	是否读出从站地址	BOOL	0：EN 复位 1：读出 Modubs 从站地址，保持
Address	Modbus 从站地址	BYTE	1—247：从站号

✧ **GET_LOCAL_ADDRESS** 功能块举例（变量定义与梯形图）



梯形图描述：

EN 置位时，该功能块使能，读取本机 Modbus 从站地址，Q 等于 1。

EN 复位时，Q 等于 0，但 Address 输出值保持原读取值。

7.2 模拟电位器功能块（Hollysys_PLC_Ex.lib）

7.2.1 POT——读取模拟电位器值

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	POT EN Q CH OUT

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
CH	模拟电位器通道号	BYTE	0：通道 0 1：通道 1
输出	**功能**	**数据类型**	**值**
Q	是否已读入数据	BOOL	0：EN 复位或数据未读入 1：第一次读入数据，保持
OUT	模拟电位器当前值	BYTE	0—255

✧ **POT 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: POT;
0005     T_OR_F: BOOL;
0006     NO0_Value: BYTE;
0007 END_VAR
```

```
0001
      EN          Example                                   T_OR_F
                  POT
     —| |——————EN      Q————————————————————————————————————( )—
             0—CH    OUT—NO0_Value
```

梯形图描述：

EN 置位时，该功能块使能，第一次读到数值后，Q 输出 1，并保持，OUT 输出的值为第 0 通道电位器的值，当电位器右旋至最大位置，OUT 值等于 255。

EN 复位时，Q 等于 0，OUT 的输出保持最后一次读入的电位器的值，不清零。

7.3 系统看门狗功能块（Hollysys_PLC_Ex.lib）

7.3.1 HD_WDT_Reset——系统看门狗复位

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	HD_WDT_RESET EN Q

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
输出	功能	数据类型	值
Q	是否复位完毕	BOOL	0：EN 复位 1：复位完毕，保持

✧ **HD_WDT_Reset 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: HD_WDT_Reset;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
        EN        Example                                  T_OR_F
                  HD_WDT_Reset
   ----| |--------|EN        Q|--------------------------( )----
```

梯形图描述:

EN 置位时，该功能块使能，看门狗复位，Q 等于 1。
EN 复位时，Q 等于 0。

注意

➢ 当用户程序特别大的时，调用改功能块。

7.4 单相计数功能块（Hollysys_PLC_Ex_CT.lib）

7.4.1 HD_CTUD_T2——T2 高速计数器

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	HD_CTUD_T2 RSTLD Q PV CV Edge_Sel UPDN EXUD_EN INT_EN

◆ 输入输出说明

外部端子输入	功能		
I0.0	外部高速计数脉冲输入		
I0.1	外部高速方向控制输入		
输入	**功能**	**数据类型**	**值**
RSTLD	使能	BOOL	0：无效 1：上升沿使能
PV	计数设定值	UINT	0—65535
Edge_Sel	I0.0 输入脉冲 触发边沿选择	BYTE	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	两个功能： 1.增/减计数选择 （EXUD_EN=0 时有效） 2.初始化时复位/装载	BOOL	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	外部 I0.1 方向控制使能 I0.1 高电平：增计数 I0.1 低电平：减计数	BOOL	0：I0.1 方向控制禁止 1：I0.1 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	BOOL	0：T2 计数值到达中断禁止 1：T2 计数值到达中断使能
输出	**功能**	**数据类型**	**值**
Q	计数值到达标志	BOOL	0：当前计数值 CV ≤ PV（UPDN=1）或者 CV ≥ 0（UPDN=0） 1：当前计数值 CV ≥ PV（UPDN=1）或者 CV ≤ 0（UPDN=0）
CV	当前计数值	UINT	0—65535

◆ 中断（计数值到达）关联事件

HD_TC2 interrupt

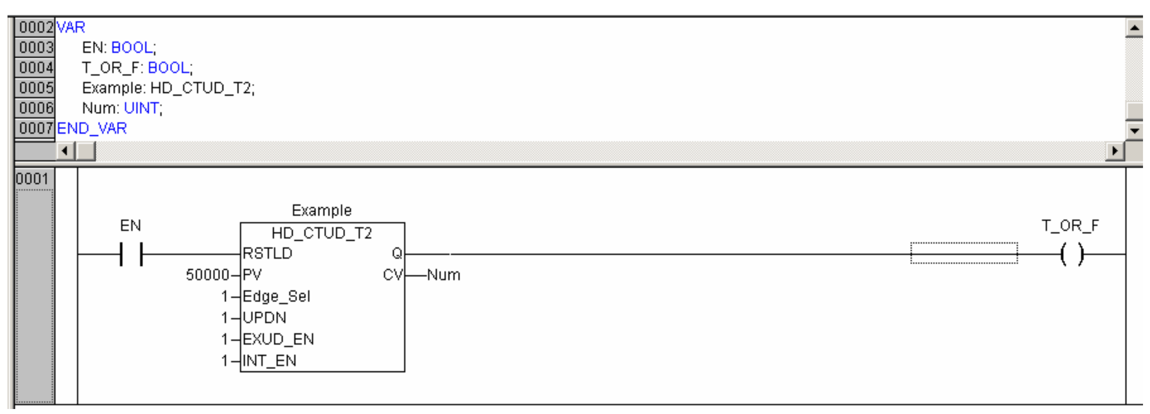
注意

- ➢ 事件使用方法参见附录 B。
- ➢ 计数频率与硬件通道性能有关。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_CTUD_T2	HD_DCTUD_T2	T2 内部计数器、 I0.0、I0.1 脉冲输入	不可使用功能块 B

✧ **HD_CTUD_T2 功能块举例（变量定义与梯形图）**



如上图设定值：

Edge_Sel = 1，I0.0 输入脉冲为上升沿时，计数一个。

UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。

EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.1 方向控制使能，I0.1 高电平时增计数，I0.1 低电平时减计数。

INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC2 interrupt 事件。

梯形图描述：

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.0 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.1 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC2 interrupt 中断事件，Q 输出 1。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.4.2 HD_CTUD_T3——T3 高速计数器

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	HD_CTUD_T3 RSTLD Q PV CV Edge_Sel UPDN EXUD_EN INT_EN

◆ 输入输出说明

外部端子输入	功能		
I0.2	外部高速计数脉冲输入		
I0.3	外部高速方向控制输入		
输入	**功能**	**数据类型**	**值**
RSTLD	使能	BOOL	0：无效 1：上升沿使能
PV	计数设定值	UINT	0—65535
Edge_Sel	I0.2 输入脉冲 触发边沿选择	BYTE	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	两个功能： 1.增/减计数选择 （EXUD_EN＝0 时有效） 2.初始化时复位/装载	BOOL	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	外部 I0.3 方向控制使能 I0.3 高电平：增计数 I0.3 低电平：减计数	BOOL	0：I0.3 方向控制禁止 1：I0.3 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	BOOL	0：T3 计数值到达中断禁止 1：T3 计数值到达中断使能
输出	**功能**	**数据类型**	**值**
Q	计数值到达标志	BOOL	0：当前计数值 CV ≤ PV（UPDN=1）或者 CV ≥ 0（UPDN=0） 1：当前计数值 CV ≥ PV（UPDN=1）或者 CV ≤ 0（UPDN=0）
CV	当前计数器值	UINT	0—65535

◆ 中断（计数值到达）关联事件

HD_TC3 interrupt

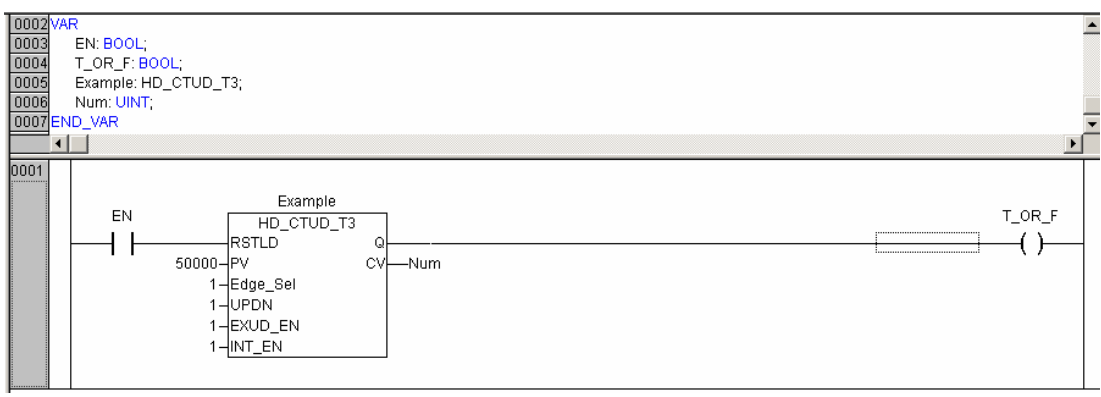
注意

- 事件使用方法参见附录 B。
- 计数频率与硬件通道性能有关。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_CTUD_T3	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		

✧ **HD_CTUD_T3** 功能块举例（变量定义与梯形图）



如上图设定值：

Edge_Sel = 1，I0.2 输入脉冲为上升沿时，计数一个。

UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。

EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.3 方向控制使能，I0.3 高电平时增计数，I0.3 低电平时减计数。

INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC3 interrupt 事件。

梯形图描述：

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.2 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.3 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC3 interrupt 中断事件，Q 输出 1。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.4.3 HD_CTUD_T4——T4 普通计数器

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	HD_CTUD_T4 RSTLD Q PV CV Edge_Sel UPDN EXUD_EN INT_EN

◆ 输入输出说明

外部端子输入	功能		
I0.4	外部普通计数脉冲输入		
I0.5	外部普通方向控制输入		
输入	**功能**	**数据类型**	**值**
RSTLD	使能	BOOL	0：无效 1：上升沿使能
PV	计数设定值	UINT	0—65535
Edge_Sel	I0.4 输入脉冲 触发边沿选择	BYTE	0：禁止 1：上升沿 2：下降沿 3：双边沿
UPDN	两个功能： 1.增/减计数选择 （EXUD_EN＝0 时有效） 2.初始化时复位/装载	BOOL	0：减计数/装载 CV=PV 1：增计数/复位 CV=0
EXUD_EN	外部 I0.5 方向控制使能 I0.5 高电平：增计数 I0.5 低电平：减计数	BOOL	0：I0.5 方向控制禁止 1：I0.5 方向控制使能 （此时 UPDN 不控制增减计数）
INT_EN	计数值到达中断使能 UPDN=0：CV=0 时产生中断 UPDN=1：CV=PV 时产生中断	BOOL	0：T4 计数值到达中断禁止 1：T4 计数值到达中断使能
输出	**功能**	**数据类型**	**值**
Q	计数值到达标志	BOOL	0:当前计数值 CV≤PV（UPDN=1）或者 CV≥0（UPDN=0） 1:当前计数值 CV≥PV（UPDN=1）或者 CV≤0（UPDN=0）
CV	当前计数器值	UINT	0—65535

◆ 中断（**T4** 溢出）关联事件

HD_TC4 interrupt

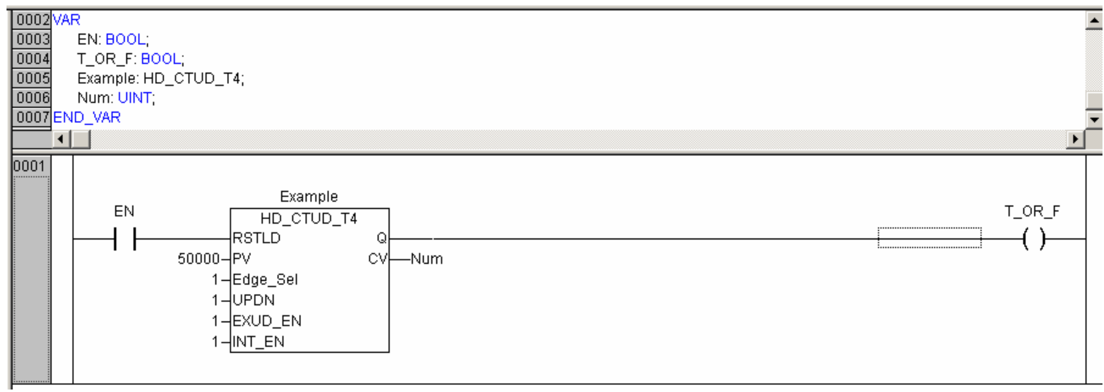
注意

➢ 事件使用方法参见附录 B。
➢ 计数频率与硬件通道性能有关。

◆ 关联冲突功能块

使用功能块 **A**	关联功能块 **B**	关联硬件	可使用程度
HD_CTUD_T4	HD_DCTUD_T4	T4 内部计数器、 I0.4、I0.5 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3	T4 内部计数器	

✧ **HD_CTUD_T4 功能块举例（变量定义与梯形图）**



如上图设定值：

Edge_Sel = 1，I0.4 输入脉冲为上升沿时，计数一个。

UPDN = 1，上电复位时 CV=0，因为此时 EXUD_EN = 1，所以 UPDN 不控制增减计数。

EXUD_EN = 1，此时 UPDN 不控制增减计数，I0.5 方向控制使能，I0.5 高电平时增计数，I0.5 低电平时减计数。

INT_EN = 1，因为 UPDN = 1，所以计数值 CV=PV 时触发中断，调用 HD_TC4 interrupt 事件。

梯形图描述:

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，UPDN 不控制增减计数，此时 I0.4 每到达一个上升沿计数一个，Num 增加或者减少 1 个（与 I0.5 外部方向控制有关），当计数值等于 50000 时，触发 HD_TC4 interrupt 中断事件，Q 输出 1。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.4.4 HD_T7_CTU——T7 高速计数器

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	HD_CTU_T7 LOAD　Q PV　CV Edge_Sel INT_EN

◆ 输入输出说明

外部端子输入	功能		
I0.6	外部高速计数脉冲输入		
输入	**功能**	**数据类型**	**值**
LOAD	使能	BOOL	0：无效 1：上升沿使能
PV	计数设定值	UINT	0—65535
Edge_Sel	I0.6 输入脉冲 触发边沿选择	BYTE	0：禁止 1：上升沿 2：下降沿 3：双边沿

INT_EN	计数值到达中断使能	BOOL	0：T7 计数设定值到达中断禁止 1：T7 计数设定值到达中断使能
输出	**功能**	**数据类型**	**值**
Q	计数值到达标志	BOOL	0：当前计数值 CV≤PV 1：当前计数值 CV≥PV
CV	当前计数器值 该功能块仅限增计数	UINT	0—65535

◆ 中断（T7 溢出）关联事件

HD_TC7 interrupt

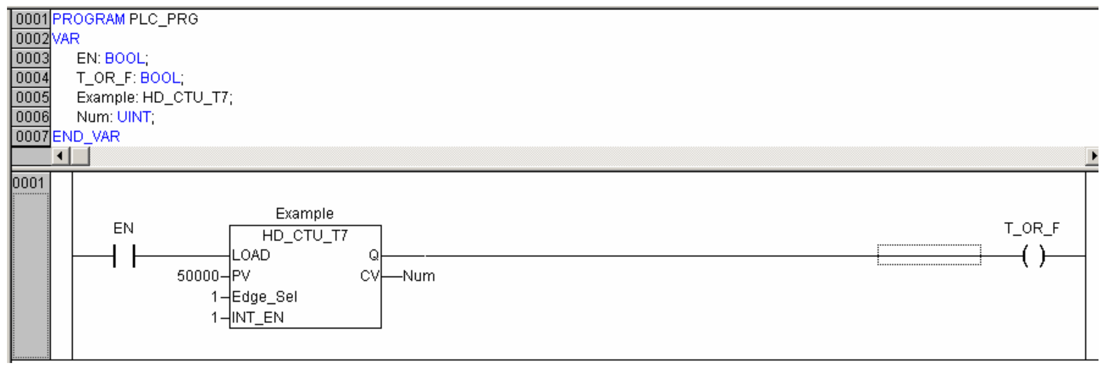
注意

➢ 事件使用方法参见附录 B。
➢ 计数频率与硬件通道性能有关。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_T7_CTU	HD_DCTUD_T2	I0.6 脉冲输入与 I0.6 外部清零脉冲输入	不可使用 B 外部清零脉冲输入功能
	Fast_ExINT Fast_ExINT_E	I0.6 脉冲输入与 I0.6 快速外部中断 3 脉冲输入	不可使用 B 快速外部中断 3 脉冲输入通道
	HD_TIMER_T7	T7	不可使用功能块 B

✧ **HD_T7_CTU 功能块举例（变量定义与梯形图）**



如上图设定值：

Edge_Sel = 1，I0.6 输入脉冲为上升沿时，计数一个。

INT_EN = 1，计数值到达标志时触发 HD_TC7 interrupt 中断事件。

梯形图描述：

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，此时 I0.6 每到达一个上升沿计数一个，Num 增加 1 个，当计数值大于等于 50000 时，触发中断，调用 HD_TC7 interrupt 事件，Q 输出 1。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.5 两相计数功能块（Hollysys_PLC_Ex_DCT.lib）

7.5.1 HD_DCTUD_T2——T2 两相高速计数器

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	HD_DCTUD_T2 EN Q PVmax CV Mode Zero_Mode

◆ 输入输出说明

外部端子输入	功能		
I0.0	外部高速计数脉冲输入 A		
I0.1	外部高速计数脉冲输入 B		
I0.6	外部清零脉冲输入		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效 1：上升沿使能
PVmax	最大计数设定值	UINT	0—65535
Mode	计数模式选择	BYTE	0：禁止 1：I0.1 的上升或下降沿 2：I0.0 的上升或下降沿 3：I0.0 或 I0.1 的上升或下降沿
Zero_Mode	清零脉冲使能（输入通道 I0.6）	BOOL	0:禁止。PVmax 有效，当 CV≥PVmax，CV 复位（I0.6 无效） 1：I0.6 清零脉冲上升沿触发 CV 复位（PVmax 无效） 2：I0.6 清零脉冲下降沿触发 CV 复位（PVmax 无效） 3:I0.6 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）
输出	**功能**	**数据类型**	**值**
Q	计数启动标志	BOOL	0：计数禁止 1：计数已启动
CV	当前计数器值	UINT	0—65535

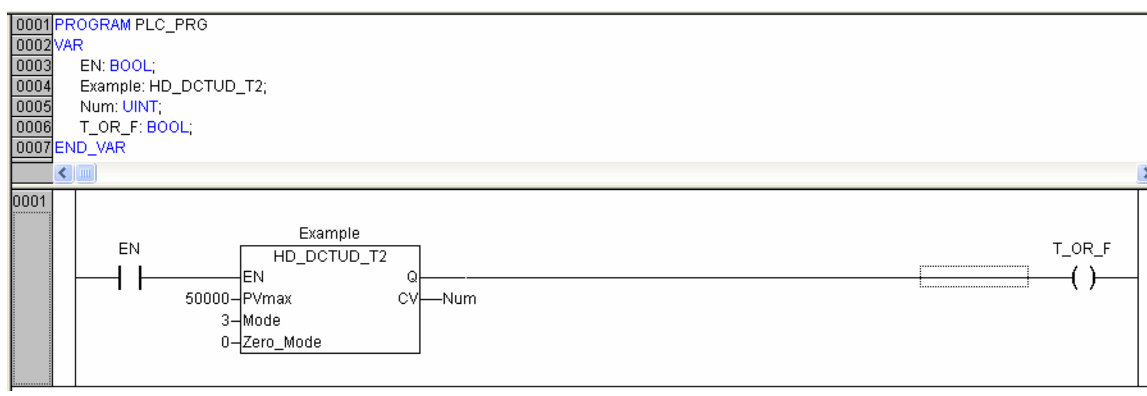
注意

- 上升或下降沿触发表示上升沿触发一次，下降沿触发一次。
- 计数频率与硬件通道性能有关。
- 两路高速脉冲输入的频率应相同，相位不同。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_DCTUD_T2	HD_CTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用功能块 B
	HD_T7_CTU	I0.6 外部清零脉冲输入与 I0.6 脉冲输入	使用 A 的外部清零脉冲输入则不可以使用 B
	Fast_ExINT	I0.6 外部清零脉冲输入与 I0.6 快速外部中断 3 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		

✧ **HD_DCTUD_T2** 功能块举例（变量定义与梯形图）



```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: HD_DCTUD_T2;
0005     Num: UINT;
0006     T_OR_F: BOOL;
0007 END_VAR
```


如上图设定值：

Mode = 3，I0.0 或 I0.1 的上升或下降沿触发计数。

Zero_Mode = 0，I0.6 外部清零脉冲输入不起作用，当 CV≥PVmax 时，CV 值为 0。

梯形图描述:

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，I0.0 或 I0.1 的上升或下降沿触发计数。当 CV≥PVmax，CV 值为 0，重新开始计数。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.5.2 HD_DCTUD_T3——T3 两相高速计数器

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	HD_DCTUD_T3 EN　Q PVmax　CV Mode Zero_Mode

◆ 输入输出说明

外部端子输入	功能
I0.2	外部高速计数脉冲输入 A
I0.3	外部高速计数脉冲输入 B
I0.7	外部清零脉冲输入

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
PVmax	最大计数设定值	UINT	0—65535
Mode	计数模式选择	BYTE	0：禁止 1：I0.3 的上升或下降沿 2：I0.2 的上升或下降沿 3：I0.2 或 I0.3 的上升或下降沿
Zero_Mode	清零脉冲使能 （输入通道 I0.7）	BOOL	0：禁止。PVmax 有效，当 CV≥PVmax，CV 复位（I0.7 无效） 1：I0.7 清零脉冲上升沿触发 CV 复位（PVmax 无效） 2：I0.7 清零脉冲下降沿触发 CV 复位（PVmax 无效） 3：I0.7 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）
输出	**功能**	**数据类型**	**值**
Q	计数启动标志	BOOL	0：计数禁止 1：计数已启动
CV	当前计数器值	UINT	0—65535

注意

- 上升或下降沿触发表示上升沿触发一次，下降沿触发一次。
- 计数频率与硬件通道性能有关。
- 两路高速脉冲输入的频率应相同，相位不同。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_DCTUD_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		
	Fast_ExINT	I0.7 外部清零脉冲输入与 I0.7 快速外部中断 2 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 2 脉冲输入
	Fast_ExINT_E		

✧ **HD_DCTUD_T3 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: HD_DCTUD_T3;
0005     Num: UINT;
0006     T_OR_F: BOOL;
0007 END_VAR
```

```
0001
          Example
        HD_DCTUD_T3
  EN                                      T_OR_F
--| |----EN          Q-------------------( )
  10000--PVmax      CV--Num
      1--Mode
      1--Zero_Mode
```

如上图设定值：

Mode = 1，I0.3 的上升或下降沿触发计数。

Zero_Mode = 1，PVmax 不起作用，当 I0.7 上升沿到达触发 CV 复位。

梯形图描述:

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，I0.3 的上升或下降沿触发计数。当 I0.7 上升沿到达触发 CV 复位，重新开始计数。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.5.3 HD_DCTUD_T4——T4 两相普通计数器

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	HD_DCTUD_T4 EN Q PVmax CV Mode Zero_Mode

◆ 输入输出说明

外部端子输入	功能		
I0.4	外部普通计数脉冲输入 A		
I0.5	外部普通计数脉冲输入 B		
I1.0	外部清零脉冲输入		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效 1：上升沿使能
PVmax	最大计数设定值	UINT	0—65535
Mode	计数模式选择	BYTE	0：禁止 1：I0.5 的上升或下降沿 2：I0.4 的上升或下降沿 3：I0.4 或 I0.5 的上升或下降沿
Zero_Mode	清零脉冲使能 （输入通道 I1.0）	BOOL	0：禁止。PVmax 有效，当 CV≥PVmax，CV 复位（I1.0 无效） 1：I1.0 清零脉冲上升沿触发 CV 复位（PVmax 无效） 2：I1.0 清零脉冲下降沿触发 CV 复位（PVmax 无效） 3：I1.0 清零脉冲上升或下降沿触发 CV 复位（PVmax 无效）
输出	**功能**	**数据类型**	**值**
Q	计数启动标志	BOOL	0：计数禁止 1：计数已启动
CV	当前计数器值	UINT	0—65535

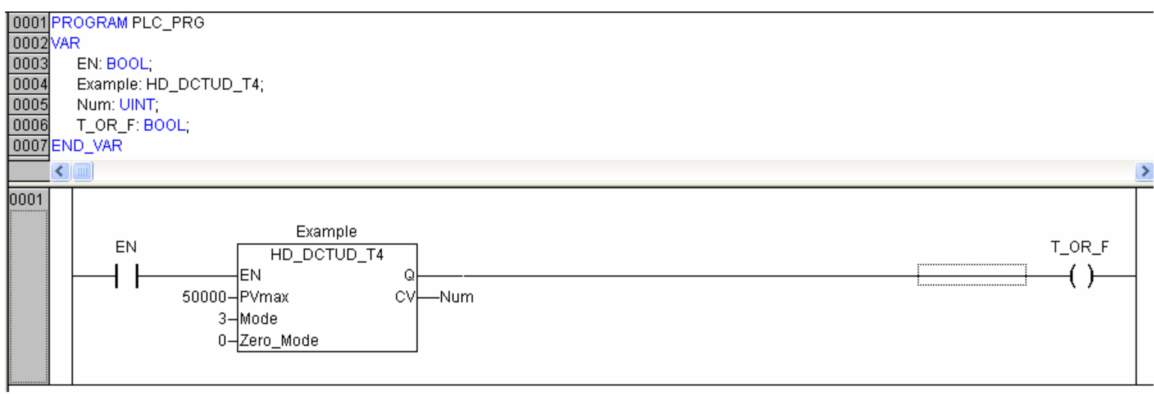
注意

- 上升或下降沿表示上升沿触发一次，下降沿触发一次。
- 计数频率与硬件通道性能有关。
- 两路普通脉冲输入的频率应相同，相位不同。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_DCTUD_T4	HD_CTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3	T4 内部计数器	
	Fast_ExINT	I1.0 外部清零脉冲输入与 I1.0 快速外部中断 1 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 1 脉冲输入通道
	Fast_ExINT_E		

✧ **HD_DCTUD_T4 功能块举例（变量定义与梯形图）**



如上图设定值：

Mode = 3，I0.4 或 I0.5 的上升或下降沿触发计数。

Zero_Mode = 0，I1.0 外部清零脉冲输入无效，当 CV≥PVmax 时，CV 值为 0。

梯形图描述：

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，I0.4 或 I0.5 的上升或下降沿触发计数。当 CV≥PVmax，CV 值为 0，重新开始计数。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.6 两相 32 位计数功能块（Hollysys_PLC_Ex_DCT32.lib）

7.6.1 HD_DCTUD32_T3——两相 32 位高速计数器

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	HD_DCTUD32_T3 EN Q Mode CV UPDN

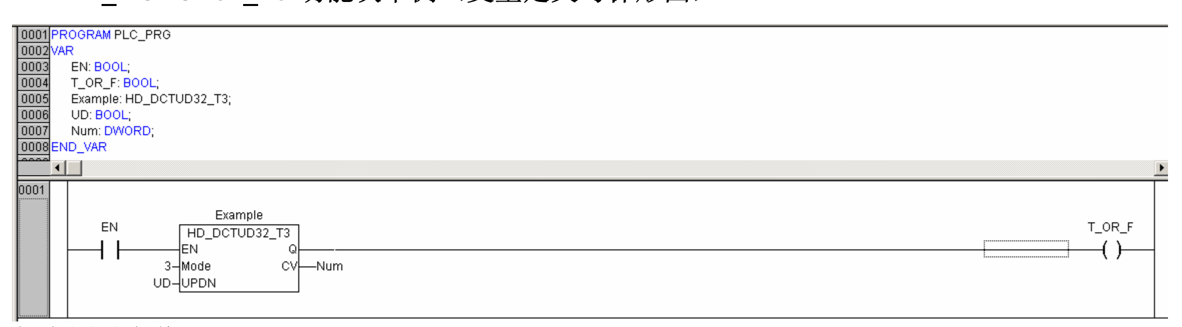
◆ 输入输出说明

外部端子输入			
I0.2	外部高速计数脉冲输入 A		
I0.3	外部高速计数脉冲输入 B		
输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
Mode	计数模式选择	BYTE	0：禁止 1：I0.3 的上升或下降沿 2：I0.2 的上升或下降沿 3：I0.2 或 I0.3 的上升或下降沿
UPDN	进位方向（低 16 位向高 16 位） （计数值-32767-32768 之间 不需要设置进位方向）	BOOL	0:向下进位 1:向上进位
输出	功能	数据类型	值
Q	计数启动标志	BOOL	0：计数禁止 1：计数已启动
CV	当前计数器值	DWORD	-2147483648——2147483647

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_DCTUD32_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用功能块 B
	HD_CTUD_T4	T4 内部计数器	
	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	
	HD_DCTUD_T4	T4 内部计数器	

✧ **HD_DCTUD32_T3 功能块举例（变量定义与梯形图）**



如上图设定值：

Mode = 3，I0.2 或 I0.3 的上升或下降沿触发计数一个。

UD 控制低 16 位向高 16 位进位方向。

梯形图描述：

EN 置位并保持时（上升沿），该功能块执行，复位计数值 CV = 0，I0.2 或 I0.3 的上升或下降沿触发计数一个，到达 32767 时如果再增计数需要保证此时 UD 等于 1，否则计数出错。

EN 复位时，停止计数，Q 等于 0，CV 值保持先前值不变。

7.7 定时器功能块（Hollysys_PLC_Ex_TIMER.lib）

7.7.1 HD_TIMER_T7——定时器

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	HD_TIMER_T7 EN　Q PERIOD

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：上升沿使能
PERIOD	触发中断间隔时间（ms）	WORD	10-2688
输出	功能	数据类型	值
Q	是否设置完毕	BOOL	0：EN 复位 1：设置完毕，保持

◆ 中断（T7 溢出）关联事件

HD_TC7 interrupt

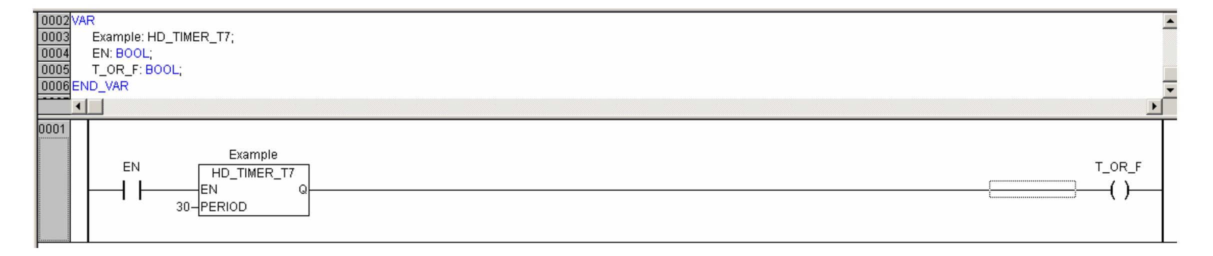
注意

➢ 事件使用方法参见附录 B。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_TIMER_T7	HD_T7_CTU	T7	不可使用功能块 B

✧ **HD_TIMER_T7** 功能块举例（变量定义与梯形图）



梯形图描述：

EN 置位并保持时（上升沿），Q 等于 1，此后每隔 30ms 就触发一次 HD_TC7 interrupt 中断事件，通过 PowerPro 中的系统事件来调用 HD_TC7 interrupt 中断事件执行程序。

EN 复位时，停止产生 HD_TC7 interrupt 中断事件，Q 等于 0。

7.8 外部中断功能块（Hollysys_PLC_Ex_ExINT.lib）

7.8.1 Fast_ExINT——快速外部中断

◆ 适用模块

存储容量为 28K 的 LM3104、LM3105、LM3106、LM3106A、LM3107。

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	FAST_EXINT EN Q Mode

◆ 输入输出说明

<table>
<tr><td>外部端子输入</td><td colspan="10">功能</td></tr>
<tr><td>I1.0</td><td colspan="10">快速外部中断 1 脉冲输入（LM3104、LM3105 无）</td></tr>
<tr><td>I0.7</td><td colspan="10">快速外部中断 2 脉冲输入</td></tr>
<tr><td>I0.6</td><td colspan="10">快速外部中断 3 脉冲输入</td></tr>
<tr><td>输入</td><td>功能</td><td>数据类型</td><td colspan="8">值</td></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td colspan="8">0：无效
1：上升沿使能</td></tr>
<tr><td rowspan="6">Mode</td><td rowspan="6">中断模式设置</td><td rowspan="6">BYTE</td><td colspan="8">Bits</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">中断 3</td><td colspan="2">中断 2</td><td colspan="2">中断 1</td><td colspan="2">无</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">中断 n 设置（n=1、2、3）</td></tr>
<tr><td colspan="8">禁止： 0 0
上升沿触发： 0 1
下降沿触发： 1 0
上升或下降沿触发：1 1</td></tr>
<tr><td>输出</td><td>功能</td><td>数据类型</td><td colspan="8">值</td></tr>
<tr><td>Q</td><td>当正确地设定参数时为 1</td><td>BOOL</td><td colspan="8">0：设置不正确
1：设置正确</td></tr>
</table>

◆ 中断（脉冲输入）关联事件

Fast External 1 interrupt　快速外部中断 1
Fast External 2 interrupt　快速外部中断 2
Fast External 3 interrupt　快速外部中断 3

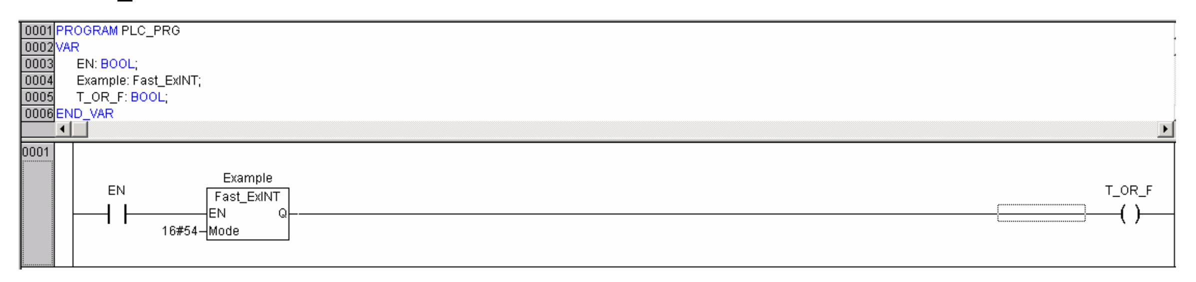
注意

➢ 事件使用方法参见附录 B。
➢ LM3104、LM3105 使用该功能块时请将快速外部中断 1 禁止。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
Fast_ExINT	HD_T7_CTU	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道、I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道

✧ **Fast_ExINT 功能块举例（变量定义与梯形图）**



如表 7-8-1，设定 Mode 值十六进制为 54（二进制为 01 01 01 00），则三个快速外部中断全部设定为上升沿触发。

表 7-8-1

Mode = 16#54							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
中断 3		中断 2		中断 1		无效	
0	1	0	1	0	1	0	0

梯形图描述:

EN 置位并保持时（上升沿），Q 等于 1，I0.6、I0.7、I1.0 每到达一个上升沿，就触发相应的快速外部中断，通过 PowerPro 的系统事件来调用相应的中断执行程序。

EN 复位时，I0.6、I0.7、I1.0 停止接收中断脉冲，Q 等于 0。

7.8.2 Fast_ExINT_E——快速外部中断

◆ 适用模块

存储容量为 120K 的 LM3106、LM3106A、LM3107、LM3108、LM3109

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	FAST_EXINT_E EN Q Mode

◆ 输入输出说明

外部端子输入	功能		
I1.1	快速外部中断 0 脉冲输入		
I1.0	快速外部中断 1 脉冲输入		
I0.7	快速外部中断 2 脉冲输入		
I0.6	快速外部中断 3 脉冲输入		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效 1：上升沿使能
Mode	中断模式设置	BYTE	Bits 7 6 | 5 4 | 3 2 | 1 0 中断 3 | 中断 2 | 中断 1 | 中断 0 中断 n 设置（n=0、1、2、3） 禁止： 0 0 上升沿触发： 0 1 下降沿触发： 1 0 上升或下降沿触发：1 1
输出	**功能**	**数据类型**	**值**
Q	当正确地设定参数时为 1	BOOL	0：设置不正确 1：设置正确

◆ 中断（脉冲输入）关联事件

Fast External 0 interrupt　快速外部中断 0
Fast External 1 interrupt　快速外部中断 1
Fast External 2 interrupt　快速外部中断 2
Fast External 3 interrupt　快速外部中断 3

注意

➢ 事件使用方法参见附录 B。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
Fast_ExINT_E	HD_T7_CTU	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入通道与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道与 I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道

✧ 变量定义与梯形图如下：

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: Fast_ExINT_E;
0005     T_OR_F: BOOL;
0006 END_VAR
```

```
0001
      EN            Example                                    T_OR_F
                    Fast_ExINT_E
 --| |-------------|EN        Q|-------------------------------( )--
            16#55--|Mode       |
```

如表 7-8-2，设定 Mode 值十六进制为 55（二进制为 01 01 01 01），四个快速外部中断全部设定为上升沿触发。

表 7-8-2

Mode = 16#55							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
中断 3		中断 2		中断 1		中断 0	
0	1	0	1	0	1	0	1

梯形图描述:

EN 置位并保持时（上升沿），Q 等于 1，I0.6、I0.7、I1.0、I1.1 每到达一个上升沿，就触发相应的快速外部中断，通过 PowerPro 的系统事件来调用相应的中断执行程序。

EN 复位时，I0.6、I0.7、I1.0、I1.1 停止接收中断脉冲，Q 等于 0。

7.9 脉冲输出功能块（Hollysys_PLC_Ex_PT.lib）

7.9.1 PTO_PWM0——PTO/PWM 脉冲输出

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	PTO_PWM0 EN　Q Mode　Pout_Number Period Duty P_Number

◆ 适用模块

LM3106、LM3106A、LM3108

◆ 输入输出说明

<table>
<tr><th>外部端子输出</th><th colspan="3">功能</th></tr>
<tr><td>Q1.1</td><td colspan="3">高速脉冲输出（Mode=0，1，2，3）</td></tr>
<tr><td>Q1.0</td><td colspan="3">高速脉冲互补输出（Mode=2，3）</td></tr>
<tr><th>输入</th><th>功能</th><th>数据类型</th><th>值</th></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td>0：无效
1：上升沿使能</td></tr>
<tr><td>Mode</td><td>脉冲输出模式选择（Mode=1、3 时 P_Number 无效，Pout_Number 输出 0）</td><td>BYTE</td><td>0：Q1.1 输出 PTO
1：Q1.1 输出 PWM
2：Q1.1 与 Q1.0 互补输出 PTO
3：Q1.1 与 Q1.0 互补输出 PWM</td></tr>
<tr><td>Period</td><td>周期（μs）设置</td><td>DWORD</td><td><table><tr><th>模块类型</th><th>PTO</th><th>PWM</th></tr><tr><td>LM3106</td><td>50-335000</td><td>50-335000</td></tr><tr><td>LM3106A</td><td>20-335000</td><td>10-335000</td></tr><tr><td>LM3108</td><td>50-335000</td><td>50-335000</td></tr></table></td></tr>
<tr><td>Duty</td><td>占空比</td><td>BYTE</td><td>0—100（PTO 模式恒为 50）</td></tr>
<tr><td>P_Number</td><td>设定脉冲个数</td><td>DWORD</td><td>0—4294967295</td></tr>
<tr><th>输出</th><th>功能</th><th>数据类型</th><th>值</th></tr>
<tr><td>Q</td><td>脉冲发送标志</td><td>BOOL</td><td>0：停止发送
1：正在发送</td></tr>
<tr><td>Pout_Number</td><td>已发出的脉冲个数</td><td>DWORD</td><td>0—4294967295</td></tr>
</table>

注意

➢ Mode、Period、Duty、P_Number 改变，输出自动改变。

◆ 关联冲突功能块

<table>
<tr><th>使用功能块 A</th><th>关联功能块 B</th><th>关联硬件</th><th>可使用程度</th></tr>
<tr><td rowspan="2">PTO_PWM0</td><td>PTO_PWM0_Run</td><td rowspan="2">T12</td><td rowspan="2">不可使用功能块 B</td></tr>
<tr><td>PTO_CYC</td></tr>
</table>

✧ **PTO_PWM0 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Example: PTO_PWM0;
0005     T_OR_F: BOOL;
0006     Num: DWORD;
0007 END_VAR
```

```
0001
                     Example
        EN           PTO_PWM0                                   T_OR_F
  ──┤ ├──────── EN              Q ──────────────────────────────( )──
            2 ── Mode  Pout_Number ── Num
          100 ── Period
           50 ── Duty
        10000 ── P_Number
```

梯形图描述:

EN 置位并保持时（上升沿），Q1.1 与 Q1.0 以相反的电平开始发送脉冲（Mode=2），脉冲的频率为 1S / 100μs = 10K（Period=100），Num 显示当前已经发送的脉冲数量，Q 等于 1 并保持，共发送 10000 个脉冲，发送完毕后停止，Q 等于 0。

发送过程中，若 EN 复位，Q1.1 与 Q1.0 停止发送脉冲，Q 等于 0，Num 保持当前值不变。

因为选择 Mode=2，输出 PTO，占空比为 50（此时不论功能块上 Duty 输入多少，占空比恒为 50 不变）。

7.9.2 PTO_PWM1——PTO/PWM 脉冲输出

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	PTO_PWM1 EN ── Q Mode ── Pout_Number Period Duty P_Number

◆ 适用模块

LM3104、LM3106、LM3106A、LM3108

◆ 输入输出说明

外部端子输出	功能		
Q0.3	高速脉冲输出（Mode=0，1）		
输入	**功能**	**数据类型**	**值**
EN	使能	BOOL	0：无效 1：上升沿使能
Mode	脉冲输出模式选择（Mode=1 时，P_Number 无效，Pout_Number 输出 0）	BYTE	0：Q0.3 输出 PTO 1：Q0.3 输出 PWM

<table>
<tr><td rowspan="5">Period</td><td rowspan="5">周期（μs）设置</td><td rowspan="5">DWORD</td><td>模块类型</td><td>PTO</td><td>PWM</td></tr>
<tr><td>LM3104</td><td>50-2630000</td><td>50-2630000</td></tr>
<tr><td>LM3106</td><td>50-2630000</td><td>50-2630000</td></tr>
<tr><td>LM3106A</td><td>20-2630000</td><td>10-2630000</td></tr>
<tr><td>LM3108</td><td>50-2630000</td><td>50-2630000</td></tr>
<tr><td>Duty</td><td>占空比</td><td>BYTE</td><td colspan="3">0—100（PTO 模式恒为 50）</td></tr>
<tr><td>P_Number</td><td>设定脉冲个数</td><td>DWORD</td><td colspan="3">0—4294967295</td></tr>
<tr><td>输出</td><td>功能</td><td>数据类型</td><td colspan="3">值</td></tr>
<tr><td>Q</td><td>脉冲发送标志</td><td>BOOL</td><td colspan="3">0：停止发送
1：正在发送</td></tr>
<tr><td>Pout_Number</td><td>已发出的脉冲个数</td><td>DWORD</td><td colspan="3">0—4294967295</td></tr>
</table>

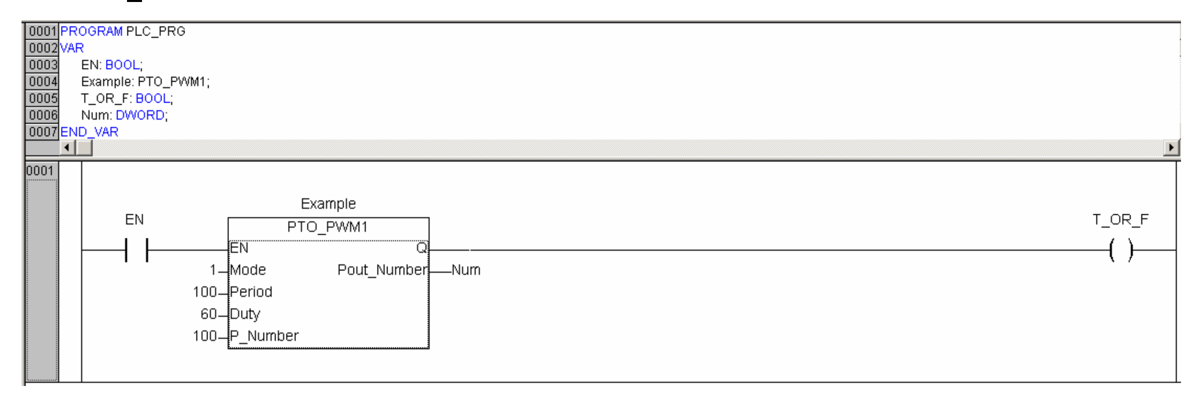
注意

- Mode、Period、Duty、P_Number 改变，输出自动改变。

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
PTO_PWM1	PTO_PWM1_Run	T8、T13	不可使用功能块 B

✧ **PTO_PWM1 功能块举例（变量定义与梯形图）**



梯形图描述:

EN 置位并保持时（上升沿），Q0.3 开始发送脉冲，脉冲的频率为 1S / 100μs = 10K，占空比为 60，Num 显示当前已经发送的脉冲数量，Q 等于 1 并保持，EN 保持 1 时不停止发送。

EN 复位时，Q0.3 停止发送脉冲，Q 等于 0。

因为选择 Mode = 1，输出 PWM，占空比为 60。

7.10 脉冲加减速输出功能块（Hollysys_PLC__EX_PTRun.lib）

7.10.1 PTO_PWM0_Run——PTO/PWM 脉冲输出（加减速）

◆ 功能块示例

POU 语言	程　序
梯形图（LD）	PTO_PWM0_RUN EN　　State Mode　　Pout_Number Start_End_Fre　　Accel_Number Run_Fre Accel_Fre Accel_Swich Duty P_Number

◆ 适用模块

LM3106、LM3106A、LM3108

◆ 功能描述

- ✓ 若 Start_End_Fre>Run_Fre，为减速运行功能，在任何模式下，则按减速—匀速运行。
- ✓ 若 Start_End_Fre<Run_Fre，为加速运行功能，在 PTO 模式下，按加速—匀速—减速运行（加速和减速过程是对称的），在 PWM 模式，按加速—匀速运行。
- ✓ 如果 Start_End_Fre=Run_Fre，则为匀速运行功能，此时 Accel_Fre、Accel_Swich 无效。

◆ 输入输出说明

<table>
<tr><th>外部端子输出</th><th colspan="5">功能</th></tr>
<tr><td>Q1.1</td><td colspan="5">高速脉冲输出（Mode=0，1，2，3）</td></tr>
<tr><td>Q1.0</td><td colspan="5">高速脉冲互补输出（Mode=2，3）</td></tr>
<tr><th>输入</th><th>功能</th><th>数据类型</th><th colspan="3">值</th></tr>
<tr><td>EN</td><td>使能</td><td>BOOL</td><td colspan="3">0：无效
1：上升沿使能</td></tr>
<tr><td>Mode</td><td>脉冲输出模式选择（Mode=1、3 时 P_Number 无效，Pout_Number 输出 0）</td><td>BYTE</td><td colspan="3">0：Q1.1 输出 PTO
1：Q1.1 输出 PWM
2：Q1.1 与 Q1.0 互补输出 PTO
3：Q1.1 与 Q1.0 互补输出 PWM</td></tr>
<tr><td rowspan="4">Start_End_Fre</td><td rowspan="4">起始频率&停止频率（HZ）</td><td rowspan="4">DWORD</td><td>模块类型</td><td>PTO</td><td>PWM</td></tr>
<tr><td>LM3106</td><td>3-20000</td><td>3-20000</td></tr>
<tr><td>LM3106A</td><td>3-50000</td><td>3-100000</td></tr>
<tr><td>LM3108</td><td>3-20000</td><td>3-20000</td></tr>
<tr><td rowspan="4">Run_Fre</td><td rowspan="4">运行频率（HZ）</td><td rowspan="4">DWORD</td><td>模块类型</td><td>PTO</td><td>PWM</td></tr>
<tr><td>LM3106</td><td>3-20000</td><td>3-20000</td></tr>
<tr><td>LM3106A</td><td>3-50000</td><td>3-100000</td></tr>
<tr><td>LM3108</td><td>3-20000</td><td>3-20000</td></tr>
</table>

Accel_Fre	加速度（HZ/S，正值）	DWORD	
Accel_Swich	频率改变开关	BOOL	0:不改变 1：改变
Duty	占空比	BYTE	0—100（PTO 模式恒为 50）
P_Number	要发送的脉冲数	DWORD	0—4294967295
输出	**功能**	**数据类型**	**值**
State	脉冲发送标志	BOOL	0：停止发送 1：正在发送
Pout_Number	已发送的脉冲数 （模式 0、2 下有效）	DWORD	0—4294967295
Accel_Number	变速阶段 发送的脉冲数	DWORD	0—4294967295

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
PTO_PWM0_Run	PTO_PWM0 PTO_CYC	T12	不可使用功能块 B

✧ **PTO_PWM0_Run 功能块举例（变量定义与梯形图）**

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Swich: BOOL;
0005     AlertTime: TON;
0006     Example: PTO_PWM0_Run;
0007     Num: DWORD;
0008     AccelNum: DWORD;
0009     T_OR_F: BOOL;
0010 END_VAR
```

```
0001
  EN    Swich        AlertTime
                       TON                                   Swich
--| |----|/|---------|IN    Q|------------------------------( )--
             T#20ms--|PT   ET|--

0002
                      Example
                    PTO_PWM0_Run                            T_OR_F
  EN
--| |--------------|EN              State|------------------( )--
                0--|Mode      Pout_Number|--Num
              400--|Start_End_Fre Accel_Number|--AccelNum
             5000--|Run_Fre
              100--|Accel_Fre
            Swich--|Accel_Swich
               50--|Duty
          1000000--|P_Number
```

梯形图描述：

EN 置位并保持时（上升沿），Q1.1 开始发送脉冲，脉冲的频率为 400HZ，Num 显示当前已发送脉冲数，AccelNum 显示 0，此后每过 20ms 脉冲频率增加 100HZ，直到脉冲频率增加到 5000HZ，AccelNum 显示加速阶段发送的脉冲数（受扫描周期影响），然后开始以 5000HZ 匀速运行，因为选择 Mode=0，所以当剩余待发送脉冲数等于加速阶段发送的脉冲数时开始减速，直到发送完 1000000 个脉冲后，停止发送，在下一个扫描周期到达时， AccelNum 清 0，Num 一直保持直到 EN 再次到达上升沿。

EN 复位时，Q1.1 停止发送脉冲，Q 等于 0。

因为选择 Mode=0，输出 PTO，占空比为 50（此时不论功能块上 Duty 等于多少，占空比恒为 50）。

7.10.2 PTO_PWM1_Run——PTO/PWM 脉冲输出（加减速）

◆ 功能块示例

POU 语言	程 序
梯形图（LD）	PTO_PWM1_RUN EN / State Mode / Pout_Number Start_End_Fre / Accel_Number Run_Fre Accel_Fre Accel_Swich Duty P_Number

◆ 适用模块

LM3104、LM3106、LM3106A、LM3108

◆ 功能描述

✓ 若 Start_End_Fre>Run_Fre，为减速运行功能，在任何模式下，则按减速—匀速运行。
✓ 若 Start_End_Fre<Run_Fre，为加速运行功能，在 PTO 模式下，按加速—匀速—减速运行（加速和减速过程是对称的），在 PWM 模式，按加速—匀速运行。
✓ 如果 Start_End_Fre=Run_Fre，则为匀速运行功能，此时 Accel_Fre、Accel_Swich 无效。

◆ 输入输出说明

外部端子输出	功能				
Q0.3	高速脉冲输出（Mode=0，1）				
输入	**功能**	**数据类型**	**值**		
EN	使能	BOOL	0：无效 1：上升沿使能		
Mode	脉冲输出模式选择（Mode=1 时，P_Number 无效，Pout_Number 输出 0）	BYTE	0：Q0.3 输出 PTO 1：Q0.3 输出 PWM		
Start_End_Fre	起始频率&停止频率（HZ）	DWORD	**模块类型**	**PTO**	**PWM**
			LM3104	1-20000	1-20000
			LM3106	1-20000	1-20000
			LM3106A	1-50000	1-100000
			LM3108	1-20000	1-20000
Run_Fre	运行频率（HZ）	DWORD	**模块类型**	**PTO**	**PWM**
			LM3104	1-20000	1-20000
			LM3106	1-20000	1-20000
			LM3106A	1-50000	1-100000
			LM3108	1-20000	1-20000
Accel_Fre	加速度（HZ，正值）	DWORD			
Accel_Swich	频率改变开关	BOOL	0:不改变 1：改变		

Duty	占空比	BYTE	0—100（PTO 模式恒为 50）
P_Number	要发送的脉冲数	DWORD	0—4294967295
输出	**功能**	**数据类型**	**值**
State	脉冲发送标志	BOOL	0：停止发送 1：正在发送
Pout_Number	已发送的脉冲数 （模式 0 下有效）	DWORD	0—4294967295
Accel_Number	变速阶段 发送的脉冲数	DWORD	0—4294967295

◆ 关联冲突功能块

使用功能块 A	关联功能块 B	关联硬件	可使用程度
PTO_PWM1_Run	PTO_PWM1	T8、T13	不可使用功能块 B

✧ **PTO_PWM1_Run 功能块举例**（变量定义与梯形图）

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     EN: BOOL;
0004     Swich: BOOL;
0005     AlertTime: TON;
0006     Example: PTO_PWM1_Run;
0007     Num: DWORD;
0008     AccelNum: DWORD;
0009     T_OR_F: BOOL;
0010 END_VAR
```

```
0001  EN --| |-- Swich --|/|-- AlertTime TON (IN, T#20ms->PT; Q, ET) ---- ( ) Swich

0002  EN --| |-- Example PTO_PWM1_Run:
          EN                    State ---- ( ) T_OR_F
          1 -Mode               Pout_Number - Num
          400 -Start_End_Fre    Accel_Number - AccelNum
          5000 -Run_Fre
          100 -Accel_Fre
          Swich -Accel_Swich
          60 -Duty
          1000000 -P_Number
```

梯形图描述:

EN 置位并保持时（上升沿），Q0.3 开始发送脉冲，脉冲的频率为 400HZ，Num 显示 0，AccelNum 显示 0。此后每过 20ms 脉冲频率增加 100HZ，直到脉冲频率增加到 5000HZ，AccelNum 显示加速阶段发送的脉冲数（受扫描周期影响），然后开始以 5000HZ 匀速运行。

EN 复位时，Q0.3 停止发送脉冲，Q 等于 0。

因为选择 Mode=1，所以 P_Number=1000000 无效，输出 PWM，占空比为设置的 60，Q0.3 将一直发送脉冲，直到 EN 等于 0。

7.10.3 步进电机升降速曲线控制方法

步进电机可以开环方式控制而无需反馈就能对位置和速度进行控制。但也正是因为负载位置对控制电路没有反馈，步进电机就必须正确响应每次励磁变化。如果励磁频率选择不当，电机不能够移到新的位置，那么实际的负载位置相对控制器所期待的位置出现永久误差，即发生失步现象或过冲现象。因此步进电机开环控制系统中，应该防止失步和过冲。

失步和过冲现象分别出现在步进电机启动和停止的时候。一般情况下，系统的极限启动频率比较低，而要求的运行速度往往比较高。如果系统以要求的运行速度直接启动，因为该速度已超过极限启动频率而不能正常启动，轻则可能发生丢步，重则根本不能启动，产生堵转。系统运行起来以后，如果达到终点时立即停止发送脉冲串，令其立即停止，则由于系统惯性作用，电机转子会转过平衡位置。如果负载的惯性很大，会使步进电机转子转到接近终点平衡位置的下一个平衡位置，并在该位置停下。

为了克服失步和过冲现象，应在步进电机启停时进行如图 7-10-1 所示的升降速控制，其中 ls 为启动频率，hs 为运行频率。

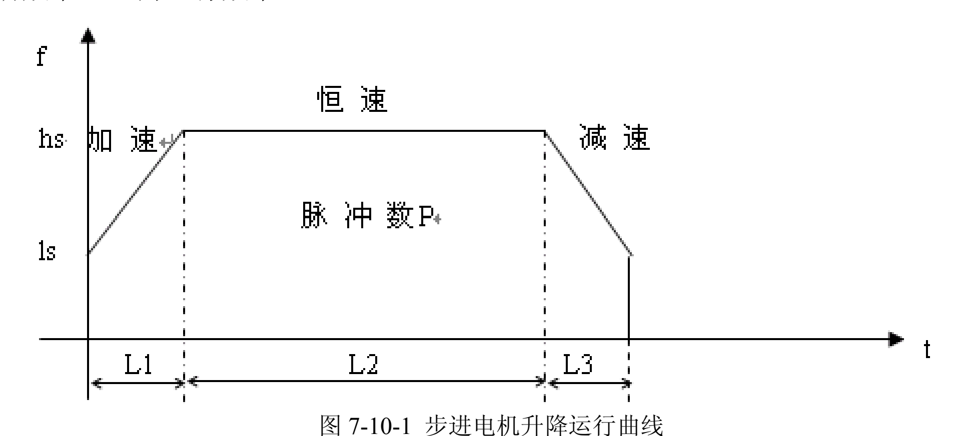


图 7-10-1 步进电机升降运行曲线

从图 7-10-1 可以看出，L2 段为恒速运行，L1 段为升频，L3 段为降频。按照“失步”的定义，如果在 L1 及 L3 段上升及下降的控制频率变化大于步进电机的响应频率变化，步进电机就会失步，失步会导致步进电机停转，经常会影响系统的正常工作。因此，在步进电机变速运行中，必须进行正确的升降速控制。

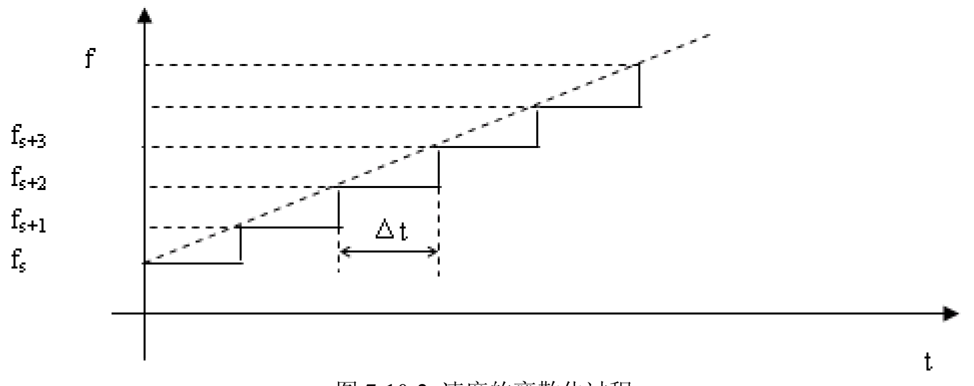


图 7-10-2 速度的离散化过程

PLC 在控制步进电机加减速的过程中，一般用离散方法逼近理想的升降速曲线。加减速的斜率在直线加速过程中，速度不是连续变化，而是按分档阶段变化，为与要求的升速斜率相逼近，必须确定每个台阶上的运行时间，见图 7-10-2。时间 Δt 越小，升速越快，反之越慢。Δt 的大小可由理论或实验确定，以升速最快而又不失步为原则。

7.11 工程量转换功能块（Hollysys_PLC_Cnvt.lib）

7.11.1　E_H——工程量转换为 16 进制数

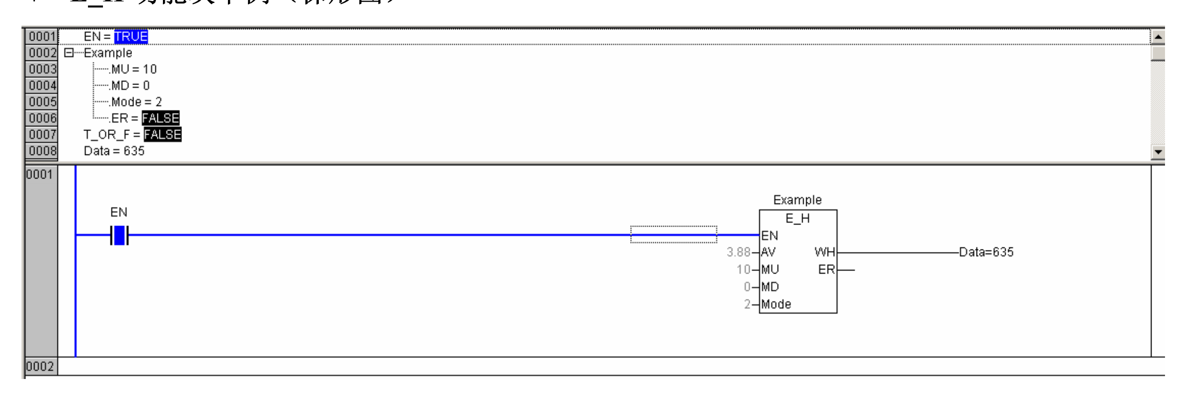
◆ 功能块示例

POU 语言	程　序
梯形图（LD）	E_H AV　WH MU　ER MD Mode

◆ 输入输出说明

输入	功能	数据类型	值
AV	工程量数据	REAL	
MU	工程量量程上限	REAL	
MD	工程量量程下限	REAL	
Mode	转换模式	BYTE	0：正常模式，MD-MU 对应 0-65535 1：电流 0-20mA/4-20mA 对应 0-3277（用于 LM3320） 2：电压 0-10V 对应 0-1638（用于 LM3320）
输出	**功能**	**数据类型**	**值**
WH	输出 16 进制数据	UINT	
ER	错误代码	BOOL	0：正确 1：错误

✧ **E_H 功能块举例（梯形图）**



梯形图描述:

EN 置位时，进行转换，工程量数据为 3.88V，输入上限是 10V，输入下限是 0V，选择模式 2，则对应于 0-1638（LM3320），此时输出值为 635。

EN 复位时，输出值 635 保持不变。

注意

➢ 必须通过使能运算符调用此功能块。

7.11.2 H_E——16 进制数转换工程量数据

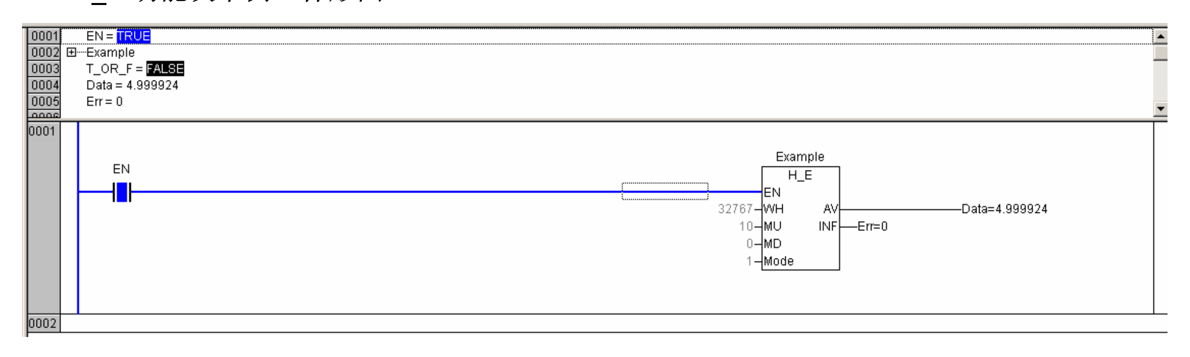
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	H_E WH AV MU INF MD Mode

◆ 输入输出说明

输入	功能	数据类型	值
WH	输入 16 进制数据	UINT	
MU	工程量量程上限	REAL	
MD	工程量量程下限	REAL	
Mode	电信号类型	BOOL	0：电流，对应 0-65535 1：电压，对应 0-65535
输出	**功能**	**数据类型**	**值**
AV	输出工程量数据	REAL	
INF	信息代码	BYTE	0：正常 1：故障

✧ **H_E 功能块举例（梯形图）**



梯形图描述:

EN 置位时，进行转换，输入数据为 32767，工程量上限是 10V，工程量下限是 0V，选择模式 1，则对应于 0-65535，此时输出值为 4.999924。

EN 复位时，输出值 4.999924 保持不变。

注意

➢ 必须通过使能运算符调用此功能块。

7.12 随机数发生功能块（Hollysys_PLC_Math.lib）

7.12.1 Rand——随机数发生

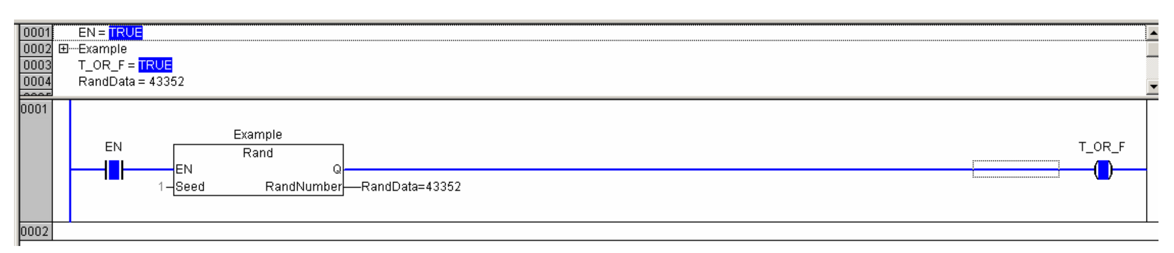
◆ 功能块示例

POU 语言	程 序
梯形图（LD）	RAND EN Q Seed RandNumber

◆ 输入输出说明

输入	功能	数据类型	值
EN	使能	BOOL	0：无效 1：使能
Seed	种子	UINT	0-65535
输出	**功能**	**数据类型**	**值**
Q	是否有效数据	BOOL	0：无效数据 1：有效数据
RandNumber	随机数	UINT	0-65535

✧ **Rand** 功能块举例（梯形图）



梯形图描述:

EN 置位时，产生随机数，每个扫描周期产生 1 个随机数。

EN 复位时，停止产生随机数，RandNumber 保持最后一次产生的随机数不变，上图为 43352。

附　录　A

A.1　PowerPro指令与功能块速查表

基本指令		
指令分类	指令定义	指令
算术运算指令	加法指令	ADD
	乘法指令	MUL
	减法指令	SUB
	除法指令	DIV
	取余指令	MOD
	索引指令	INDEXOF
	数据类型大小指令	SIZEOF
赋值运算指令	赋值指令	MOVE
布尔运算指令	与指令	AND
	或指令	OR
	异或指令	XOR
	取非指令	NOT
移位运算指令	左移指令	SHL
	右移指令	SHR
	循环左移指令	ROL
	循环右移指令	ROR
选择运算指令	二选一指令	SEL
	取最大值指令	MAX
	取最小值指令	MIN
	极限值指令	LIMIT
	多选一指令	MUX
比较运算指令	大于指令	GT
	小于指令	LT
	大于等于指令	GE
	小于等于指令	LE
	等于指令	EQ
	不等于指令	NE

数据类型转换指令	布尔类型转换指令	BOOL_TO_<TYPE>	
	字节类型转换指令	BYTE_TO_<TYPE>	
	日期类型转换指令	DATE_TO_<TYPE>	
	长整型转换指令	DINT_TO_<TYPE>	
	日期时间类型转换指令	DT_TO_<TYPE>	
	双字类型转换指令	DWORD_TO_<TYPE>	
	整数类型转换指令	INT_TO_<TYPE>	
	字类型转换指令	WORD_TO_<TYPE>	
	实数类型转换指令	REAL_TO_<TYPE>	
	短整型转换指令	SINT_TO_<TYPE>	
	字符类型转换指令	STRING_TO_<TYPE>	
	时钟类型转换指令	TIME_TO_<TYPE>	
	时间类型转换指令	TOD_TO_<TYPE>	
	无符号长整型转换指令	UDINT_TO_<TYPE>	
	无符号整型转换指令	UINT_TO_<TYPE>	
	无符号短整型转换指令	USINT_TO_<TYPE>	
	截短转换指令	TRUNC	
初等数学运算指令	绝对值指令	ABS	
	平方根指令	SQRT	
	自然对数指令	LN	
	常用对数指令	LOG	
	指数指令	EXP	
	正弦指令	SIN	
	余弦指令	COS	
	正切指令	TAN	
	反正弦指令	ASIN	
	反余弦指令	ACOS	
	反正切指令	ATAN	
	幂指令	EXPT	
地址运算指令	取地址指令	ADR	
	取地址内容指令	^	
调用运算指令	调用运算指令	CAL	
G3 指令			
指令分类	**指令定义**	**指令**	**所在库**
字符串功能	结合字符串指令	CONCAT	Standard.lib
	删除字符指令	DELETE	
	查找字符串指令	FIND	
	插入字符串指令	INSERT	
	左边取字符串指令	LEFT	

字符串功能	字符串长度指令	LEN	
	中间取字符串指令	MID	
	替换字符串指令	REPLACE	
	右边取字符串指令	RIGHT	
BCD 转换	BCD 码转整型指令	BCD_TO_INT	Util.lib
	整型转 BCD 码指令	INT_TO_BCD	
位/字节功能	位提取指令	EXTRACT	Util.lib
	位整合指令	PACK	
	位赋值指令	PUTBIT	
	位拆分指令	UNPACK	
库版本信息	读取库版本查看	Version_Util	Util.lib
版本信息	读取软件版本信息	SyslibGetVersion2300	SysLibC16x.lib
事件	事件调用	SysCallbackRegister	SysLibCallBack.lib
	解除事件调用	SysCallbackUnregistrer	
PowerPro 内部功能块			
功能块分类	**功能块定义**	**功能块**	**所在库**
高等数学运算功能块	微分	DERIVATIVE	Util.lib
	积分	INTEGRAL	
	整型统计	STATISTIC_INT	
	实型统计	STATISTIC_REAL	
	平方偏差	VARIANCE	
PID 控制器功能块	比例控制器	P	Util.lib
	比例微分控制器	PD	
	比例积分微分控制器	PID	
信号发生器功能块	脉冲信号发生器	BLINK	Util.lib
	典型周期信号发生器	GEN	
函数操纵器功能块	特征曲线	CHARCURVE	Util.lib
	整型限速	RAMP_INT	
	实型限速	RAMP_REAL	
模拟量处理功能块	滞后	HYSTERESIS	Util.lib
	上下限报警	LIMITALARM	
PID2 运算功能块	PID 运算	PID2	Hollysys_PLC_Util.lib
Modbus 校验功能块	Modbus 校验产生	Generate_CRC	Hollysys_PLC_Modbus_CRC.lib
SFC 动作控制功能块	SFC 动作控制	SFCActionControl	Iecsfc.lib
PowerPro 外部 G3 功能块			
功能块分类	**功能块定义**	**功能块**	**所在库**
双稳态功能块	置位双稳态功能块	SR	Standard.lib
	复位双稳态功能块	RS	
	软件信号灯	SEMA	
触发器功能块	上升沿检测触发器	R_TRIG	Standard.lib
	下降沿检测触发器	F_TRIG	

计数器功能块	递增计数器	CTU	Standard.lib
	递减计数器	CTD	
	递增递减计数器	CTUD	
定时器功能块	普通定时器	TP	Standard.lib
	通电延时定时器	TON	
	断电延时定时器	TOF	
	实时时钟	RTC	
模拟量模块处理功能块	模拟量输入功能块	Analog_IN	Hollysys_PLC_Analog.lib
	模拟量输出功能块	Analog_OUT	
RS232 自由口通讯功能块	RS232 自由口通讯参数设置	Set_COMM_PRMT	Hollysys_PLC_Comm.lib
	RS232 自由口通讯数据发送	COMM_SEND	
	RS232 自由口通讯数据接收	COMM_RECEIVE	
	RS 232 恢复 Hollysys 专有协议	Reset_COMM_PRMT	
RS485 自由口通讯功能块	RS485 自由口通讯参数设置	Set_COMM2_PRMT	Hollysys_PLC_Comm2.lib
	RS485 自由口通讯数据发送	COMM2_SEND	
	RS485 自由口通讯数据接收	COMM2_RECEIVE	
	RS485 恢复 Hollysys 专有协议	Reset_COMM2_PRMT	
Profibus-DP 功能块	Profibus-DP 从站功能块（LM3401）	DP_Slave	Hollysys_PLC_DPSlave.lib
以太网功能块	以太网功能块（LM3403）	EtherNet_TCP	Hollysys_PLC_EtherNet.lib
硬件实时时钟功能块	设置实时时钟（DT 数据格式）	Set_HD_RTC	Hollysys_PLC_HdRtc.lib
	设置实时时钟（普通数据格式）	Set_HD_RTC_X	
	读取实时时钟日期/时间/星期	Get_HD_RTC	
实时时钟报警功能块	读取硬件实时时钟报警时间/星期	Get_HDRTC_ALM	Hollysys_PLC_HdRtcALM.lib
	设置硬件实时时钟报警时间/星期	Set_HDRTC_ALM	
自设定组脉冲发送功能块	自设定组脉冲循环发送	PTO_CYC	Hollysys_PLC_PTCYC.lib
PowerPro 外部扩展功能块			
功能块分类	功能块定义	功能块	所在库
Modubs 功能块	设置本机 Modbus 从站通讯地址	SET_LOCAL_ADDRESS	Hollysys_PLC_Ex.lib
	读取本机 Modbus 从站通讯地址	GET_LOCAL_ADDRESS	
模拟电位器功能块	读取模拟电位器值	POT	Hollysys_PLC_Ex.lib
系统看门狗功能块	系统看门狗复位	HD_WDT_Reset	Hollysys_PLC_Ex.lib

单相计数功能块	T2 高速计数器	HD_CTUD_T2	Hollysys_PLC_Ex_CT.lib
	T3 高速计数器	HD_CTUD_T3	
	T4 普通计数器	HD_CTUD_T4	
	T7 高速计数器	HD_T7_CTU	
两相计数功能块	T2 两相高速计数器	HD_DCTUD_T2	Hollysys_PLC_Ex_DCT.lib
	T3 两相高速计数器	HD_DCTUD_T3	
	T4 两相普通计数器	HD_DCTUD_T4	
两相 32 位计数功能块	32 位高速计数器	HD_DCTUD32_T3	Hollysys_PLC_Ex_DCT32.lib
定时器功能块	定时器	HD_TIMER_T7	Hollysys_PLC_Ex_TIMER.lib
外部中断功能块	快速外部中断（LM3106、LM3107）	Fast_ExINT	Hollysys_PLC_Ex_ExINT.lib
	快速外部中断（LM3108、LM3109）	Fast_ExINT_E	
脉冲输出功能块	PTO/PWM 脉冲输出	PTO_PWM0	Hollysys_PLC_Ex_PT.lib
		PTO_PWM1	
脉冲加减速输出功能块	PTO/PWM 脉冲输出（加减速）	PTO_PWM0_Run	Hollysys_PLC__EX_PTRun.lib
		PTO_PWM1_Run	
工程量转换功能块	工程量转换为 16 进制数	E_H	Hollysys_PLC_Cnvt.lib
	16 进制数转换工程量数据	H_E	
随机数发生功能块	随机数发生	Rand	Hollysys_PLC_Math.lib

A.2 PowerPro功能块关联冲突速查表

部分功能块因为使用了同一内部硬件，或者使用了相同的输入/输出端口，所以不可以同时使用。下表列出使用功能块 A 时，与其相关联的所有功能块 B，并列出功能块 B 的可使用程度。

使用功能块 A	关联功能块 B	关联硬件	可使用程度
HD_CTUD_T2	HD_DCTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用功能块 B
HD_CTUD_T3	HD_DCTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		
HD_CTUD_T4	HD_DCTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		
HD_T7_CTU	HD_DCTUD_T2	I0.6 脉冲输入与 I0.6 外部清零脉冲输入	不可使用 B 外部清零脉冲输入功能
	Fast_ExINT	I0.6 脉冲输入与 I0.6 快速外部中断 3 脉冲输入	不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		
	HD_TIMER_T7	T7	不可使用功能块 B
HD_DCTUD_T2	HD_CTUD_T2	T2 内部计数器、I0.0、I0.1 脉冲输入	不可使用功能块 B
	HD_T7_CTU	I0.6 外部清零脉冲输入与 I0.6 脉冲输入	使用 A 的外部清零脉冲输入则不可以使用 B
	Fast_ExINT	I0.6 外部清零脉冲输入与 I0.6 快速外部中断 3 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 3 脉冲输入通道
	Fast_ExINT_E		
HD_DCTUD_T3	HD_CTUD_T3	T3 内部计数器、I0.2、I0.3 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		
	Fast_ExINT	I0.7 外部清零脉冲输入与 I0.7 快速外部中断 2 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 2 脉冲输入
	Fast_ExINT_E		
HD_DCTUD_T4	HD_CTUD_T4	T4 内部计数器、I0.4、I0.5 脉冲输入	不可使用功能块 B
	HD_DCTUD32_T3		
	Fast_ExINT	I1.0 外部清零脉冲输入与 I1.0 快速外部中断 1 脉冲输入	使用 A 的外部清零脉冲输入，则不可使用 B 快速外部中断 1 脉冲输入通道
	Fast_ExINT_E		
Fast_ExINT	HD_T7_CTU	I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B
	HD_DCTUD_T2	I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入	使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T3	I0.7 快速外部中断 2 脉冲输入与 I0.7 外部清零脉冲输入	使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道
	HD_DCTUD_T4	I1.0 快速外部中断 1 脉冲输入通道、I1.0 外部清零脉冲输入	使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道

<table>
<tr><td rowspan="4">Fast_ExINT_E</td><td>HD_T7_CTU</td><td>I0.6 快速外部中断 3 脉冲输入与 I0.6 脉冲输入</td><td>使用 A 快速外部中断 3 脉冲输入，则不可使用 B</td></tr>
<tr><td>HD_DCTUD_T2</td><td>I0.6 快速外部中断 3 脉冲输入与 I0.6 外部清零脉冲输入</td><td>使用 A 快速外部中断 3 脉冲输入，则不可使用 B 外部清零脉冲输入通道</td></tr>
<tr><td>HD_DCTUD_T3</td><td>I0.7 快速外部中断 2 脉冲输入通道与 I0.7 外部清零脉冲输入</td><td>使用 A 快速外部中断 2 脉冲输入，则不可使用 B 外部清零脉冲输入通道</td></tr>
<tr><td>HD_DCTUD_T4</td><td>I1.0 快速外部中断 1 脉冲输入通道与 I1.0 外部清零脉冲输入</td><td>使用 A 快速外部中断 1 脉冲输入，则不可使用 B 外部清零脉冲输入通道</td></tr>
<tr><td rowspan="2">PTO_PWM0</td><td>PTO_PWM0_Run</td><td rowspan="2">T12</td><td rowspan="2">不可使用功能块 B</td></tr>
<tr><td>PTO_CYC</td></tr>
<tr><td>PTO_PWM1</td><td>PTO_PWM1_Run</td><td>T8、T13</td><td>不可使用功能块 B</td></tr>
<tr><td rowspan="2">PTO_PWM0_Run</td><td>PTO_PWM0</td><td rowspan="2">T12</td><td rowspan="2">不可使用功能块 B</td></tr>
<tr><td>PTO_CYC</td></tr>
<tr><td>PTO_PWM1_Run</td><td>PTO_PWM1</td><td>T8、T13</td><td>不可使用功能块 B</td></tr>
<tr><td rowspan="2">PTO_CYC</td><td>PTO_PWM0</td><td rowspan="2">T12</td><td rowspan="2">不可使用功能块 B</td></tr>
<tr><td>PTO_PWM0_Run</td></tr>
<tr><td rowspan="4">HD_DCTUD32_T3</td><td>HD_CTUD_T3</td><td>T3 内部计数器、
I0.2、I0.3 脉冲输入</td><td rowspan="4">不可使用功能块 B</td></tr>
<tr><td>HD_CTUD_T4</td><td>T4 内部计数器</td></tr>
<tr><td>HD_DCTUD_T3</td><td>T3 内部计数器、
I0.2、I0.3 脉冲输入</td></tr>
<tr><td>HD_DCTUD_T4</td><td>T4 内部计数器</td></tr>
<tr><td>HD_TIMER_T7</td><td>HD_T7_CTU</td><td>T7</td><td>不可使用功能块 B</td></tr>
</table>

A.3 硬件模块状态信息

在 M 区中，偏移从 0 开始的 100 个字节（即%MB0~%MB99）被系统占用，主要用来反映 CPU 模块及扩展模块的一些状态信息。其中%MB0~%MB9 为 CPU 模块的专用寄存器，%MB10~%MB19 保留，从%MB20 开始为模块的诊断区。

A-3-1 专用寄存器区

%MB0~%MB9 的 10 个字节，为专用寄存器，其中%MB0、%MB1 为错误标志，%MB2、%MB3 表示错误数据，%MB2 为数据低 8 位，%MB3 为数据高 8 位。

%MB0：

高位 低位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0=1：扩展地址 0 模块通讯 CRC 校验错，错误数据无定义。
Bit1=1：扩展地址 1 模块通讯 CRC 校验错，错误数据无定义。
Bit2=1：扩展地址 2 模块通讯 CRC 校验错，错误数据无定义。
Bit3=1：扩展地址 3 模块通讯 CRC 校验错，错误数据无定义。
Bit4=1：扩展地址 4 模块通讯 CRC 校验错，错误数据无定义。
Bit5=1：扩展地址 5 模块通讯 CRC 校验错，错误数据无定义。
Bit6=1：扩展地址 6 模块通讯 CRC 校验错，错误数据无定义。

%MB1：

高位 低位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0=1：SSC 硬件操作错，系统故障，错误数据为当前 SSCCON。
Bit1=1：ASC 硬件操作错，错误数据为当前 S0CON 值。
Bit2=1：ADC 硬件操作错，错误数据为当前 ADEIC 值。
Bit3=1：B 类硬件错，错误数据为当前 TFR 值。
Bit4=1：系统栈下溢错，错误数据为当前 TFR 值。
Bit5=1：系统栈上溢错，错误数据为当前 TFR 值。
Bit6=1，系统 NMI 错，错误数据为当前 TFR 值。

%MB2、%MB3：错误数据。

高字节 低字节

%MB3	%MB2

A-3-2 模块诊断区

采用 PowerPro2.0 软件编程进行硬件模块配置时，为每个模块（包括 CPU 模块）在 M 区分配了 10 字节的诊断区。CPU 模块的诊断区从%MB20 开始，扩展模块的诊断区从%MB30 开始，每个模块占 10 字节。每个模块的起始地址可以从 PLC 配置中模块的基本参数项中读取。如图 A-3-1 所示。

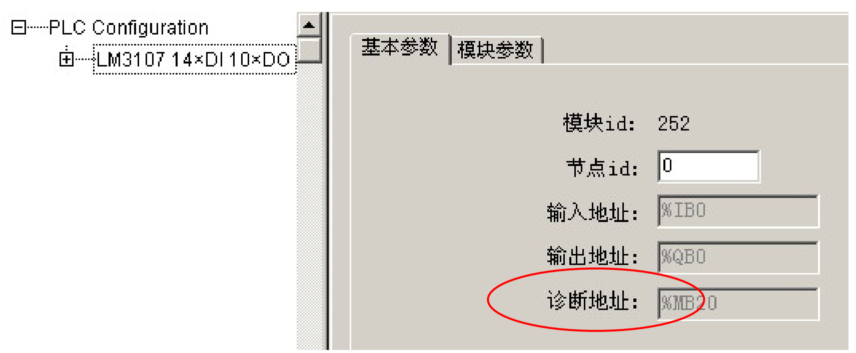


图 A-3-1

模块诊断字节的第 1 字节表示模块状态，不同的模块有不同的模块状态；对于模拟量输入模块从第 2 字节开始表示通道状态，每个通道占用 4bit, 低通道占用低 4 位，高通道占用高 4 位。

下面详细说明每种模块的模块状态定义：

◆ CPU 模块诊断区

CPU 模块：只有 1 字节为模块状态。

Bit7							Bit0
×	×	×	×	×	×	×	ID

Bit0=1，模块 ID 配置正确； Bit0=0，模块 ID 配置错。

Bit1~Bit7：保留。

第 2~10 字节保留。

◆ 开关量模块诊断区

开关量模块：

第 1 字节：

Bit7							Bit0
×	RD	×	×	×	×	×	ID

Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。

Bit6=1，CPU 模块读取扩展模块 ID 成功；Bit6=0，表示读取扩展模块 ID 不成功。

Bit1~Bit5、Bit7：保留。

第 2 字节为第 1 字节中 ID 正确的情况下，读取到的扩展模块 ID，否则为 0。

第 3~10 字节保留。

◆ 模拟量输入模块诊断区

模拟量输入模块：

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
WR	RD	×	AD_OK	READY	CHANNEL	PWR	ID

Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。

Bit1=1，外供电 DC 24V 电源正常，Bit1=0，无外供 DC 24V 电源。

Bit2=1，无通道错误，Bit2=0，有通道错误。

Bit3=1，模块已经准备好，Bit3=0，模块没有准备好。

Bit4=1，AD 芯片正常，Bit4=0，AD 芯片异常。

Bit5：保留。

Bit6=1，CPU 模块读配置数据正常，Bit6=0，CPU 模块读配置数据错误。

Bit7=1，CPU 模块写配置数据正常，Bit7=0，CPU 模块写配置数据错误。

第 2 字节以后为通道状态：

每 4 位表示一个通道的状态信息，不同的模块类型有不同的通道状态信息。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
第 2 通道（高通道）				第 1 通道（低通道）			

LM3310 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

0x01：断线（仅仅对 4-20mA 量程时有效）。

0x2：通道没有准备好。

0xF：通道无错误。

其他：保留。

LM3311 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

0x1：断偶。

0x2：通道没有准备好。

0xF：通道无错误。

其他：保留。

LM3312 通道状态定义，4 通道，占用 2、3 字节，其余字节保留：

0x0：超量程，电阻值过小（<39Ω）。

0x2：通道没有准备好。

0xF：通道无错误。

其它：保留。

LM3314 通道状态定义，8 通道，占用 2~5 字节，其余字节保留：

0x0：超量程，电阻值过小（<600Ω）。

0x1：超量程，电阻值过大（>100KΩ）。

0x2：通道没有准备好。

0xF：通道无错误。

其它：保留。

◆ **模拟量输出模块诊断区**

模拟量输入模块：

Bit7	6	5	4	3	2	1	Bit0
WR	RD	×	×	×	×	PWR	ID

Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。

Bit1=1，外供电 DC 24V 电源正常，Bit1=0，无外供 DC 24V 电源。

Bit2~5：保留。

Bit6=1，CPU 模块读配置数据正常，Bit6=0，CPU 模块读配置数据错误。

Bit7=1，CPU 模块写配置数据正常，Bit7=0，CPU 模块写配置数据错误。

模拟量输出模块没有通道状态，字节 2~10 保留。

◆ **DP 模块诊断区**

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
WR	RD	×	SPC3_STATE	OUTDATA_LEN	INDATA_LEN	×	ID

Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。

Bit1：保留。

Bit2=1，PowerPro 中对 DP 模块配置的输入数据字节数与 DP-Master 组态软件中对 DP 模块配置的输出数据字节数匹配；Bit2=0，上述字节数不匹配。

Bit3=1，PowerPro 中对 DP 模块配置的输出数据字节数与 DP-Master 组态软件中对 DP 模块

配置的输入数据字节数匹配；Bit3=0，上述字节数不匹配。

Bit4=1，SPC3 通讯正常；Bit4=0，SPC3 通讯错误。

Bit5：保留。

Bit6=1，CPU 模块读配置数据正常，Bit6=0，CPU 模块读配置数据错误。

Bit7=1，CPU 模块写配置数据正常，Bit7=0，CPU 模块写配置数据错误。

DP 模块无通道状态，字节 2~10 保留。

◆ 以太网模块诊断区

第 1 字节：模块状态。

Bit7	6	5	4	3	2	1	Bit0
WR	RD	×	×	×	×	SWP	ID

Bit0=1，模块 ID 配置正确；Bit0=0，模块 ID 配置错。

Bit1=1，有网络数据交换，Bit1=0 无网络数据交换。

Bit2~Bit5：保留。

Bit6=1，CPU 模块读配置数据正常，Bit6=0，CPU 模块读配置数据错误。

Bit7=1，CPU 模块写配置数据正常，Bit7=0，CPU 模块写配置数据错误。

以太网模块没有通道状态，字节 2~10 保留。

附　录　B

B.1　电机循环启动发送脉冲举例

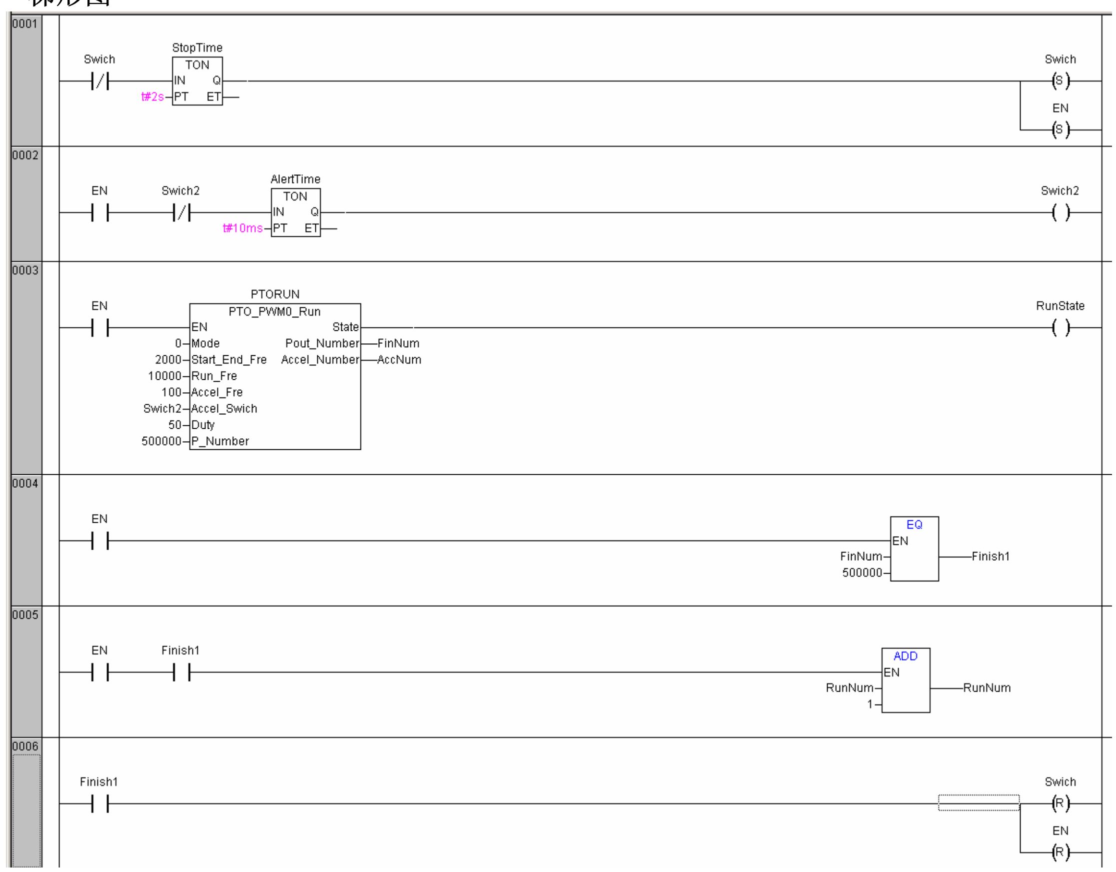
◆　要求

步进电机以 2KHz 的速度启动，以 10KHz 的速度运行，发送 500000 个脉冲，然后停止 2 秒钟，再以 2KHz 的速度启动，以 10KHz 的速度运行，发送 500000 个脉冲，如此重复下去。

◆　定义变量

```
PROGRAM PLC_PRG
    VAR
        Swich: BOOL;
        Swich2: BOOL;
        EN: BOOL;
        Finish1: BOOL;
        RunNum: WORD;
        RunState: BOOL;
        FinNum: DWORD;
        AccNum: DWORD;
        AlertTime: TON;
        PTORUN: PTO_PWM0_Run;
        StopTime: TON;
    END_VAR
```

◆　梯形图



◆ 描述

当程序开始运行时，2 秒后电机以 2KHz 的频率启动，以后每隔 10ms 增加 100Hz，当增加至 10KHz 后开始匀速运行，当剩余未发送脉冲数等于加速阶段发送的脉冲数时，开始减速运行，减速曲线与加速曲线对应，当 500000 个脉冲发送完毕，电机停止运行 2 秒钟，然后重新开始重复上述过程。

B.2 Profibus-DP功能块使用举例

◆ 要求

PLC 通过 DP 从模块向 DP 主模块发送 8 个字节的数据，同时从 DP 主模块接收 8 个字节的数据。

◆ 变量定义

```
PROGRAM PLC_PRG
    VAR
        EN: BOOL;
        Example: DP_Slave;
        T_OR_F: BOOL;
        SendDataA: WORD;   PLC 向 DP 主站发送的数据 A
        SendDataB: WORD;   PLC 向 DP 主站发送的数据 B
        SendDataC: WORD;   PLC 向 DP 主站发送的数据 C
        SendDataD: WORD;   PLC 向 DP 主站发送的数据 D
        RecDataA: WORD;    DP 主站向 PLC 发送的数据 A
        RecDataB: WORD;    DP 主站向 PLC 发送的数据 B
        RecDataC: WORD;    DP 主站向 PLC 发送的数据 C
        RecDataD: WORD;    DP 主站向 PLC 发送的数据 D
    END_VAR
```

◆ 软件配置

➢ 配置3401DP从模块如图B-2-1中鼠标位置处所示。
InputDataLen_Byte 为 DP 主站向 PLC 发送的数据长度，输入接收的字节数 8；
OutputDataLen_Byte 为 PLC 向 DP 主站发送的数据长度，输入发送的字节数 8。

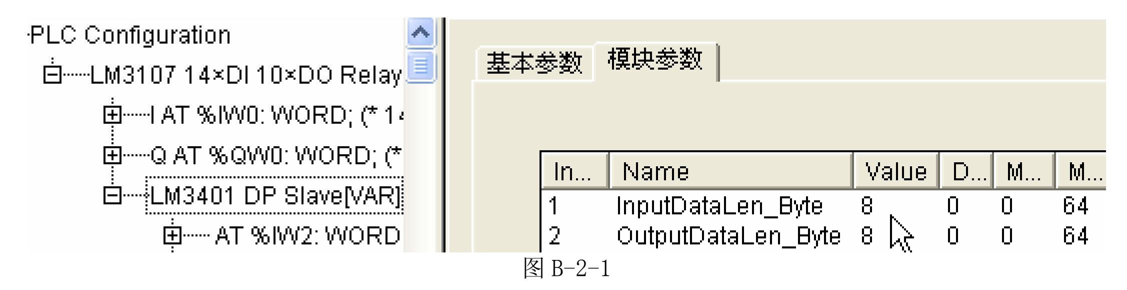


图 B-2-1

➢ DP_Slave功能块中的Address与图B-2-2所示的节点id一致，DP主站向PLC发送的数据A、B、C、D分别存放在下图所示的%IW2、%IW4、%IW6、%IW8之中。

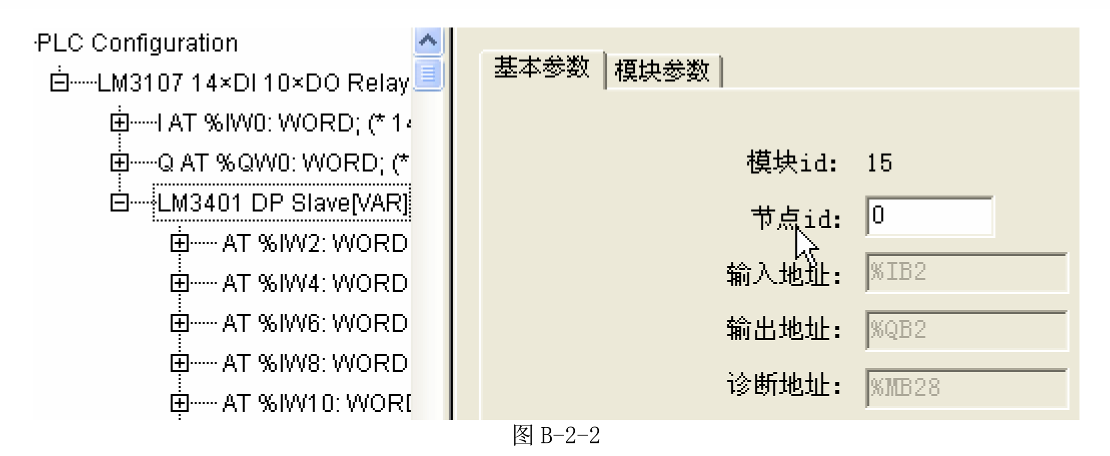


图 B-2-2

➢ PLC向DP主站发送的数据A、B、C、D分别存放在图B-2-3所示的%QW2、%QW4、%QW6、%QW8之中。

AT %IW64: WORD; (* channel 32 *) [CHANNEL (I)]
AT %QW2: WORD; (* channel 33 *) [CHANNEL (Q)]
AT %QW4: WORD; (* channel 34 *) [CHANNEL (Q)]
AT %QW6: WORD; (* channel 35 *) [CHANNEL (Q)]
AT %QW8: WORD; (* channel 36 *) [CHANNEL (Q)]
AT %QW10: WORD; (* channel 37 *) [CHANNEL (Q)]

图 B-2-3

➢ 配置DP主站的接收和发送区，DP从站的QW区数据会自动传送到DP主站的接收区，DP主站的发送区数据会自动传送到DP从站的IW区。

◆ 梯形图

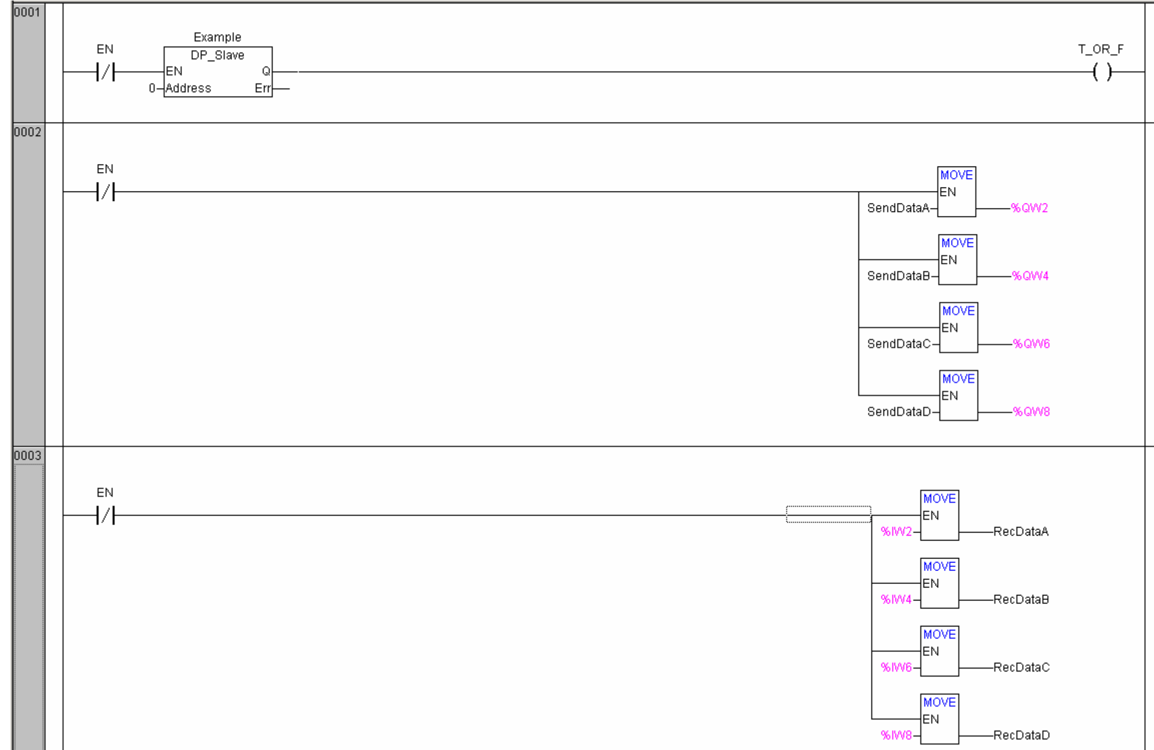


图 B-2-4

◆ 描述

上电后，程序不停地将 SendDataA、SendDataB、SendDataC、SendDataD 的数据拷贝到%QW2、%QW4、%QW6、%QW8 之中，%QW2、%QW4、%QW6、%QW8 之中的数据会自动发送到 DP 主站的接收区。

DP 主站发送的数据会自动存放在%IW2、%IW4、%IW6、%IW8 之中，程序不断的把%IW2、%IW4、%IW6、%IW8 之中的数据分别拷贝到 RecDataA、RecDataB、RecDataC、RecDataD 之中。

B. 3 以太网功能块使用举例

本例程以组态王 V6.5 为例，PC 机（MODBUS/TCP 主站）通过以太网模块向 PLC 输入区发送 2 个字的数据，从 PLC 输出区接收 2 个字的数据，同时 PC 机按位操作向 PLC 输入区发送 1 个位，从 PLC 输出区接收 1 个位。我们首先介绍在 PowePro 中如何通过组态对 LM3403 进行操作，然后简单介绍一下 MODBUS/TCP 主站的组态方法（以组态王 V6.5 为例）。

● PowerPro 程序

◆ 变量定义

```
PROGRAM PLC_PRG
    VAR
        EN: BOOL;
        Example: EtherNet_TCP;
        T_OR_F: BOOL;
        SendDataA: WORD;   (*PLC 向 MODBUS/TCP 主站发送的数据 A*)
        SendDataB: WORD;   (*PLC 向 MODBUS/TCP 主站发送的数据 B*)
        SendBitC: BOOL;    (*PLC 向 MODBUS/TCP 主站发送的位 C*)
        RecDataA: WORD;    (*MODBUS/TCP 主站向 PLC 发送的数据 A*)
        RecDataB: WORD;    (*MODBUS/TCP 主站向 PLC 发送的数据 B*)
        RecBitC: BOOL;     (*MODBUS/TCP 主站向 PLC 发送的位 C*)
    END_VAR
```

◆ 软件配置

➢ 配置LM3403以太网模块如图B-3-1。



图 B-3-1

✓ IP_Address 为本以太网模块 IP 地址（必须与 PC 机在同一网段，且无冲突的 IP 地址）。
✓ Subnet_Mask 为子网掩码，与 PC 机的子网掩码一致。
✓ GateWay_Addres 为网关地址。

- ✓ MAC_Address 不填。
- ✓ ReadDataLen_Byte 为 PC 机向 PLC 发送的数据长度，输入接收的字节数 8（必须大于实际用到的长度，且最大 200）。
- ✓ WriteDataLen_Byte 为 PLC 向 PC 机发送的数据长度，输入发送的字节数 8（必须大于实际用到的长度，且最大 200）。

➢ 以太网功能块中的Address与图B-3-2所示的节点id一致，PC机向PLC发送的数据A、B分别存放在下图所示的%IW4、%IW6，位C存放在%IX8.0。

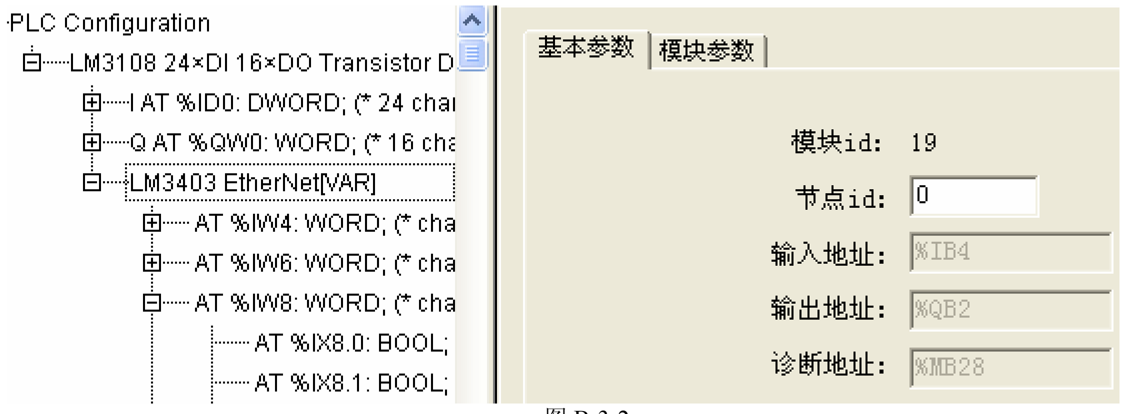


图 B-3-2

➢ PLC向PC机发送的数据A、B分别存放在图B-3-3所示的%QW2、%QW4之中， 位C存放在%QX6.0。

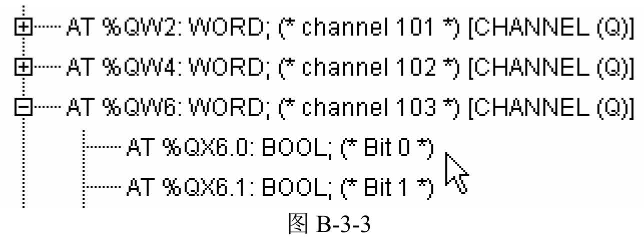


图 B-3-3

◆ 梯形图



图 B-3-4

● 组态王程序

这里给出新建设备和数据词典中关键几步，其他画面按默认或查看组态软件相关帮助。

◆ 新建设备

1. 在工程浏览器中新建设备如图 B-3-5，选中 Modbus（以太网）-网卡。

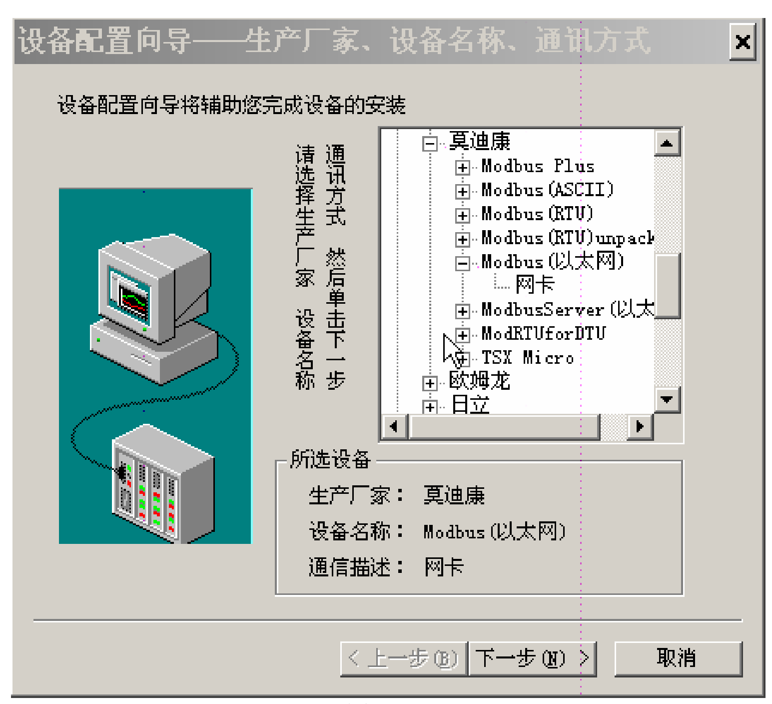


图 B-3-5

2. 选择“串口号”，选择一个不存在且与其他不冲突的串口号，如图 B-3-6：



图 B-3-6

3. 如图 B-3-7 输入设备地址，首先是从站 IP 地址与 PowerPro 中的一致，然后是空格，输入从站编号，如果不清楚可以点击地址帮助查看。

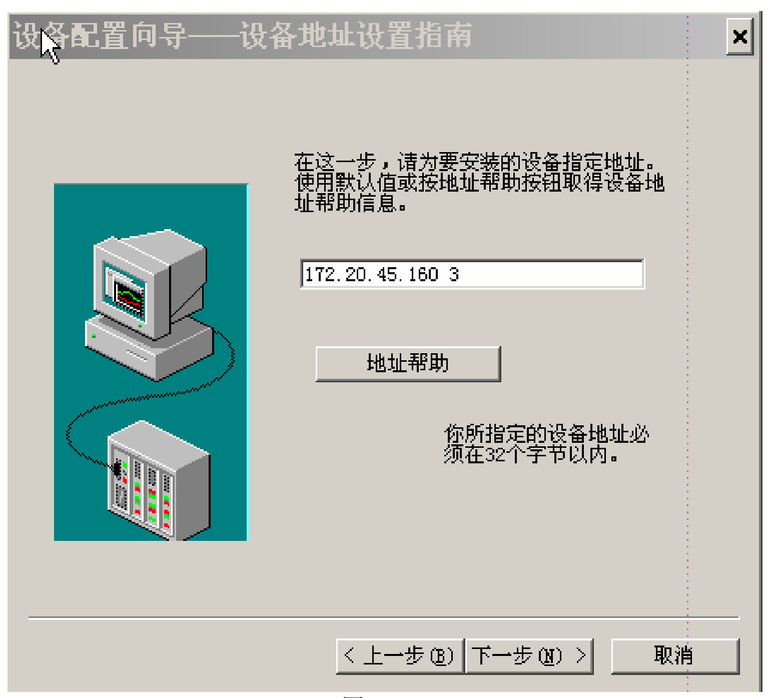


图 B-3-7

◆ 新建数据词典

1. 如图 B-3-8，变量类型选择 I/O 整数，寄存器 30001 对应 PLC 输出区第 1 个字（整型即上面组态的%QW2）。

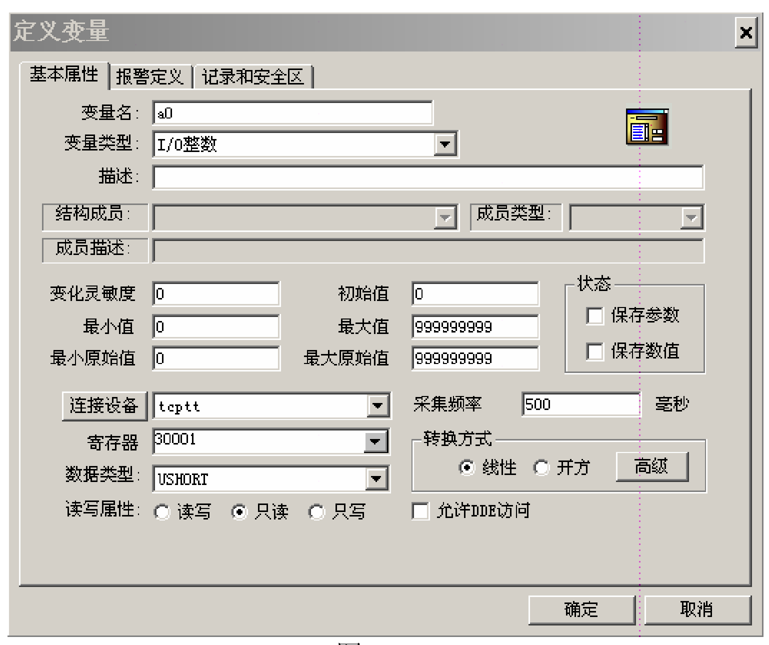


图 B-3-8

2. 如图 B-3-9，变量类型选择 I/O 整型，寄存器 30002 对应 PLC 输出区第 2 个字（整型即上面组态的%QW4，因为 PowerPro 的序号是以字节来标记的）。同理，寄存器 30003 对应 PLC 输出区第 3 个字（整型即上面组态的%QW6），以此类推。

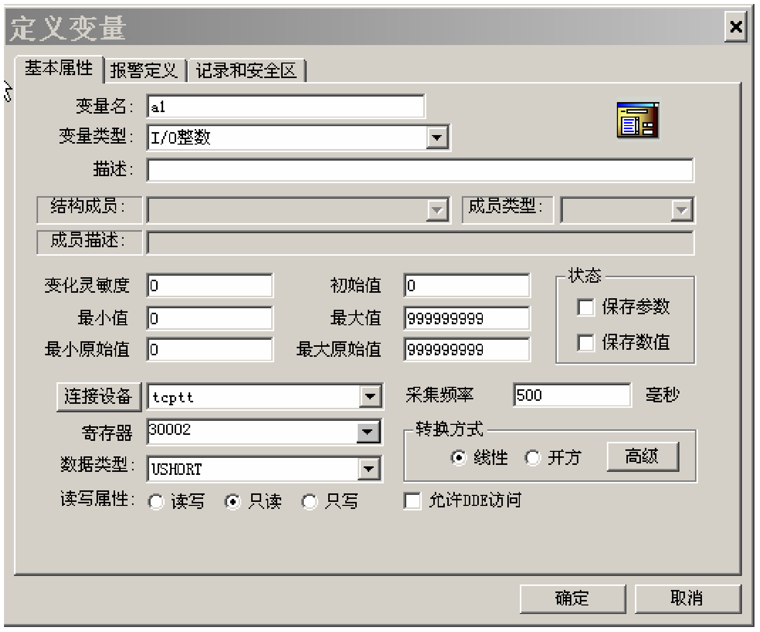


图 B-3-9

3. 如图 B-3-10，变量类型选择 I/O 整数，寄存器 40001 对应 PLC 输入区第 1 个字（整型即上面组态的%IW4）。

定义变量
基本属性 报警定义 记录和安全区
变量名: b0
变量类型: I/O整数
描述:
结构成员:
成员类型:
成员描述:
变化灵敏度 0
初始值 0
最小值 0
最大值 999999999
最小原始值 0
最大原始值 999999999
状态
保存参数
保存数值
连接设备 tcptt
采集频率 500 毫秒
寄存器 40001
转换方式
线性 开方 高级
数据类型: USHORT
读写属性: 读写 只读 只写
允许DDE访问
确定 取消

图 B-3-10

4. 如图 B-3-11，变量类型选择 I/O 整数，寄存器 40002 对应 PLC 输入区第 2 个字（整型即上面组态的%IW6）。同理，寄存器 40003 对应 PLC 输入区第 3 个字（整型即上面组态的%IW8），以此类推。



图 B-3-11

5. 如图 B-3-12，变量类型选择 I/O 离散型，寄存器 10033 对应 PLC 输出区第 33 个位（即上面组态的%QX6.0）。



图 B-3-12

6. 如图 B-3-13，变量类型选择 I/O 离散型，寄存器 00033 对应 PLC 输入区第 33 个位（即上面组态的%IX8.0）。



图 B-3-13

B.4 中断关联事件使用举例

◆ 中断事件优先级顺序

中断事件	优先级顺序
Fast External 0 interrupt	低
Fast External 1 interrupt	↓
Fast External 2 interrupt	↓
Fast External 3 interrupt	↓
HD_TC7 interrupt	↓
HD_TC2 interrupt	↓
HD_TC3 interrupt	↓
HD_TC4 interrupt	↓
HD_RTC_ALM 0 interrupt	高

◆ 要求

不受 PLC 扫描周期的影响，I0.6 每到达一个上升沿，PLC 立刻响应该脉冲，产生一个中断，%MW100 中的值增加 10。I0.7 每到达一个上升沿，PLC 立刻响应该脉冲，产生一个中断，%MW102 中的值增加 15。

◆ 程序分析

按上述要求，硬件可使用 LM3106 CPU 模块，软件需要使用以下功能块：

- Fast_ExINT（快速外部中断）

程序分为如下 3 部分：

- 主程序——定义 3106 的快速外部中断模式
- INT3PRO——I0.6 脉冲到达执行的中断程序——%MW100 中的值增加 10
- INT2PRO——I0.7 脉冲到达执行的中断程序——%MW102 中的值增加 15

◆ 程序

主程序：

1. 首先要将程序所用到的库 Hollysys_PLC_Ex_ExINT.lib 添加到库管理器，如何添加库请参见 1.5 节。

2. 按照要求，I0.6、I0.7 每到达一个上升沿产生中断，I1.0 不使用，则 Fast_ExINT 功能块的 Mode=16#50，主程序变量定义与梯形图如图 B-4-1。

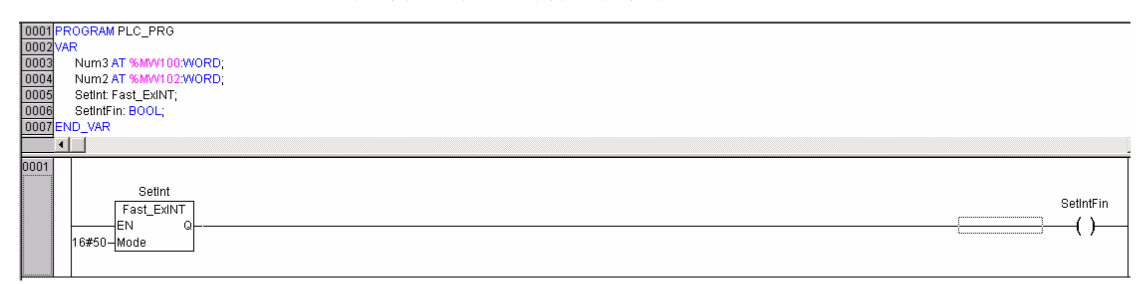


图 B-4-1

3. 由于我们使用了快速外部中断 3（I0.6）、快速外部中断 2（I0.7），所以在图 B-4-2 鼠标所示位置前打勾。

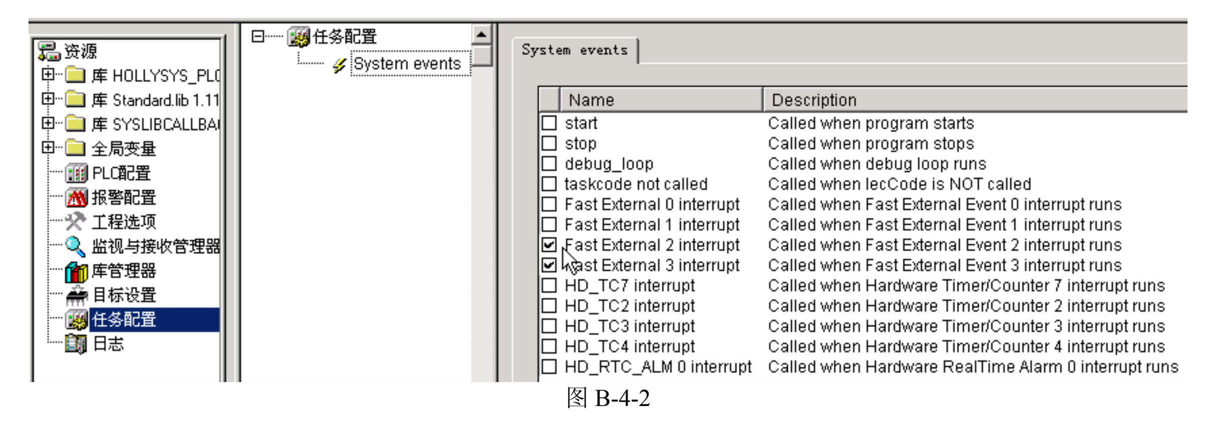


图 B-4-2

4. 在 Fast Extennal 2 interrupt 和 Fast Extennal 3 interrupt 后面分别创建一个子程序 INT2Pro、INT3Pro，分别点击 Create POU，则 2 个中断程序被创建成功，如图 B-4-3。

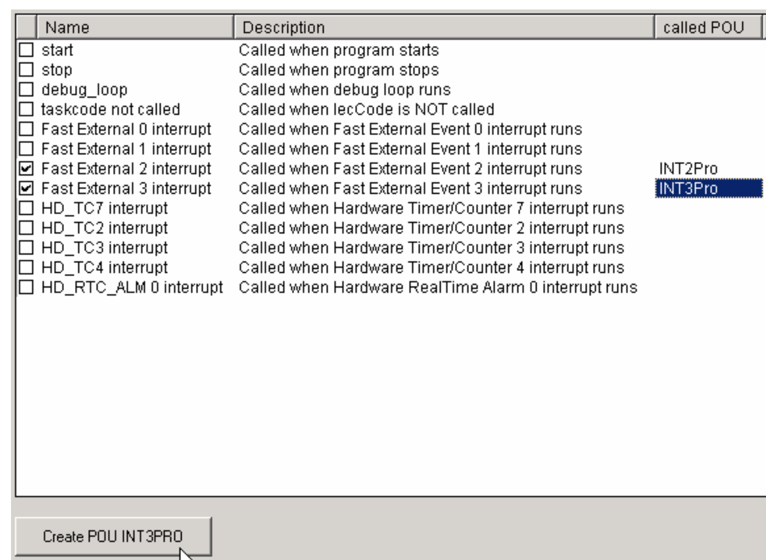


图 B-4-3

5. 创建的中断程序，默认为 ST 语言，可以选择转换为 LD，转换前需要编译通过，如图 B-4-4。

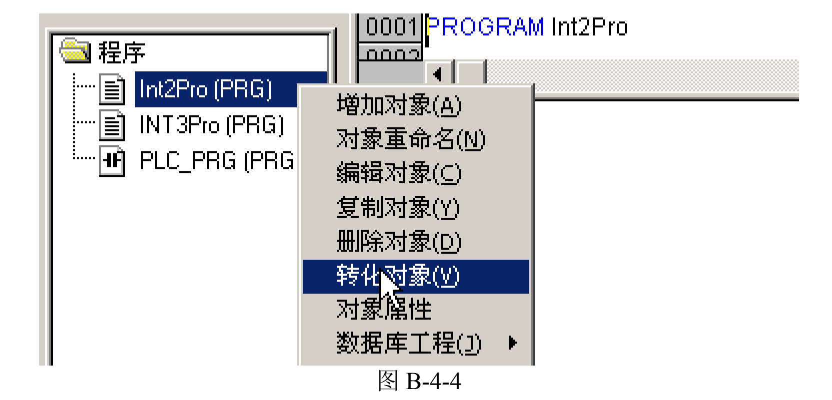


图 B-4-4

6. INT3Pro 中断程序梯形如图 B-4-5。

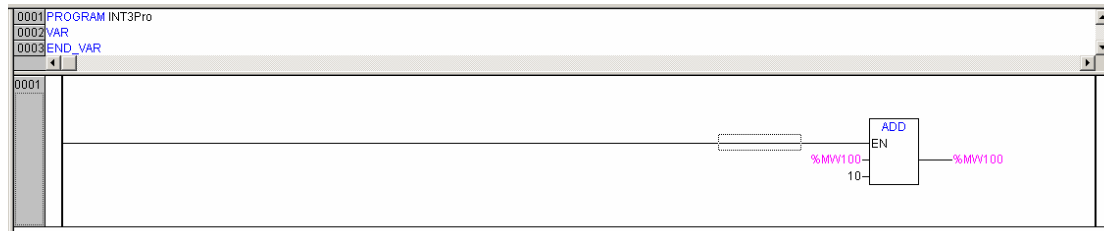


图 B-4-5

7. INT2Pro 中断程序梯形图如图 B-4-6。

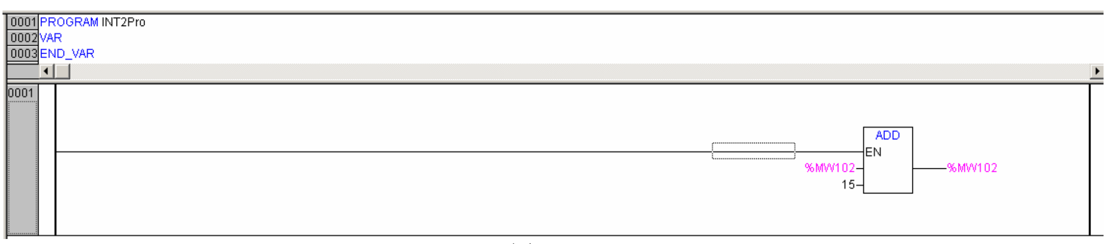


图 B-4-6

B.5 自由口通讯使用举例

◆ 要求

一检测仪表与 LM3108 模块进行 485 自由口通讯，仪表发出的协议格式如下：

自由协议格式（共 **18** 个字节）																	
STX	A	B	C	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	CR	CKS
起始符（02H）	状态字 ABC			重量						皮重						回车 0DH	校验

状态字 A			
Bit2	Bit1	Bit0	小数点位置
0	0	0	XXXX00
0	0	1	XXXXX0
0	1	0	XXXXXX
0	1	1	XXXXX.X
1	0	0	XXXX.XX
1	0	1	XXX.XXX
1	1	0	XX.XXXX
1	1	1	X.XXXXX

LM3108 需要解析的部分是:状态字 A 中的前三位，也就是 bit 0、bit 1、bit 2 表示后面“重量”处数据的小数点位置，另外就是“重量”所对应的具体的数字。仪表发送 0～10 之间的数值，LM3108 进行解析后输出 0～10V 的模拟量信号。

通讯参数为：9600Bps，8 位，无校验。

◆ 程序

变量定义

```
PROGRAM PLC_PRG
    VAR
        RecFin: BOOL;
        Set485: Set_COMM2_PRMT;
```

```
        RecData AT %MB300 : ARRAY [1..18] OF BYTE;
        RightData AT %MB320 : ARRAY [1..18] OF BYTE;
        Set485Sw: BOOL;
        Set485Fin: BOOL;
        SetRecSw: BOOL;
        RecM: COMM2_RECEIVE;
        StartQ: BOOL;
        EndQ: BOOL;
        copyArr: copy_Arr;
        L5Dot: BOOL;
        L4Dot: BOOL;
        L3Dot: BOOL;
        L2Dot: BOOL;
        L1Dot: BOOL;
        L0Dot: BOOL;
        R1Dot: BOOL;
        R2Dot: BOOL;
        ResNum: DWORD;
        AddAllNum: AddNum;
        ResNumTemp: DWORD;
        DIVNUM: DWORD;
        OUTNUMH AT %MW202 : WORD;
        OUTNUML AT %MW200 : WORD;
        OUTNUM AT %MD200 : DWORD;
        AnOutInit: BOOL;
        AnOut: Analog_OUT;
        AnOutFin: BOOL;
    END_VAR
```

◆ 主程序梯形图

0001
设定3108自由口参数,9600Bps，8位，无校验
Set485Sw
Set485
Set_COMM2_PRMT
EN Q
16#88 Mode
Set485Fin

0002
3108不停的接收数据存放于RecData
SetRecSw RecFin
RecM
COMM2_RECEIVE
EN Q
18 Number
300 TBL
RecFin

0003
判断接收数据的头尾是否正确
RecFin
EQ
EN
%MB300
16#02
StartQ
EQ
EN
%MB316
16#0D
EndQ

0004

头尾正确的话则将接收到的数据RecData 拷贝数组到RightData

RecFin StartQ EndQ

copyArr
copy_Arr
EN
RecData Arr_Source Arr_Des RightData
18 CopyLen FINISH

0005

解析状态字A，将其高5位置0，只保留低3位

RecFin copyArr.FINISH

AND
EN
%MB321 %MB321
7

0006

判断状态字A的低3位的状态,以判断小数点的位置

RecFin copyArr.FINISH

EQ
EN
%MB321 R2Dot
0

EQ
EN
%MB321 R1Dot
1

EQ
EN
%MB321 L0Dot
2

EQ
EN
%MB321 L1Dot
3

EQ
EN
%MB321 L2Dot
4

EQ
EN
%MB321 L3Dot
5

EQ
EN
%MB321 L4Dot
6

EQ
EN
%MB321 L5Dot
7

0007

将"重量"处的每个字节的数拼成一个DWORD型的数据

RecFin copyArr.FINISH — AddAllNum (AddNum) EN, Result — ResNum

0008

转换成模拟量输出模块需要的数据0-1638,小数点位置在下面程序标定

RecFin copyArr.FINISH — MUL EN; ResNum, 1638 — ResNumTemp

0009

判断小数点的位置,输出数据

RecFin copyArr.FINISH L5Dot — DIV EN; ResNumTemp, 1000000 — OUTNUM

0010

RecFin copyArr.FINISH L4Dot — DIV EN; ResNumTemp, 100000 — OUTNUM

0011

RecFin copyArr.FINISH L3Dot — DIV EN; ResNumTemp, 10000 — OUTNUM

0012

RecFin copyArr.FINISH L2Dot — DIV EN; ResNumTemp, 1000 — OUTNUM

0013

RecFin copyArr.FINISH L1Dot — DIV EN; ResNumTemp, 100 — OUTNUM

0014

RecFin copyArr.FINISH L0Dot — DIV EN; ResNumTemp, 10 — OUTNUM

```
0015
RecFin      copyArr.FINISH      R1Dot                         MOVE
                                                              EN
                                                ResNumTemp-        -OUTNUM
0016
RecFin      copyArr.FINISH      R2Dot                         MUL
                                                              EN
                                                ResNumTemp-        -OUTNUM
                                                        10-
0017
RecFin      copyArr.FINISH                                    MOVE
                                                              EN
                                                  OUTNUML-         -%QW2
0018
                  AnOut
AnOutInit         Analog_OUT                                  AnOutFin
  -|/|-           EN        Q ----------------------------------( )
                0-Address
```

◆ **AddNum** 自定义功能块程序

```
0001 FUNCTION_BLOCK AddNum
0002 VAR_INPUT
0003 END_VAR
0004 VAR_OUTPUT
0005     Result: DWORD;
0006 END_VAR
0007 VAR
0008 END_VAR
```

```
0001
0002 Result := %MB124*100000+%MB125*10000+%MB126*1000+%MB127*100+%MB128*10+%MB129;
```

◆ **copy_Arr** 自定义功能块程序

```
0001 FUNCTION_BLOCK copy_Arr
0002 VAR
0003     i: WORD;
0004 END_VAR
0005 VAR_INPUT
0006     Arr_Source: ARRAY [1..18] OF BYTE;
0007     CopyLen: WORD;
0008 END_VAR
0009 VAR_OUTPUT
0010     Arr_Des: ARRAY [1..18] OF BYTE;
0011     FINISH: BOOL := FALSE;
0012 END_VAR
```

```
0001 FOR i:=1 TO CopyLen BY 1 DO
0002     Arr_Des[i]:=Arr_Source[i];
0003 END_FOR;
0004 FINISH:=TRUE;
```